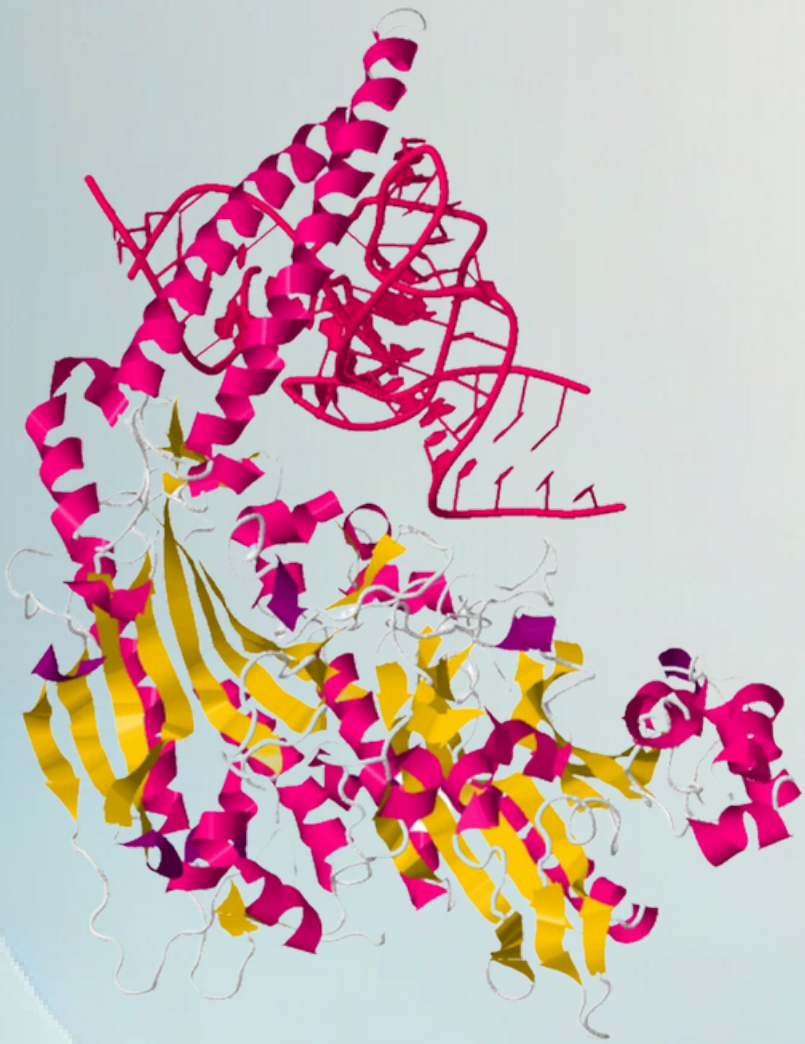


Bioquímica Quantitativa & R

José Maurício Schneedorf Ferreira da Silva

À minha família...

Disponível em: *https://bioquanti.netlify.app/*
Capa e editoração : do autor
ISBN: 978-65-00-52125-2

Endereço: Rua Gabriel Monteiro da Silva, 700 Centro – Alfenas – Minas Gerais – Brasil – CEP: 37.130-001.
Reitor: Sandro Amadeu Cerveira.
Vice-reitor: Alessandro A. Costa Pereira.

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Silva, José Maurício Schneedorf Ferreira da
Bioquímica quantitativa & R [livro eletrônico] / José Maurício Schneedorf Ferreira da Silva. -- Alfenas, MG : Ed. do Autor, 2022.
PDF

Bibliografia.
ISBN 978-65-00-52125-2

1. Biologia molecular 2. Bioquímica 3. Bioquímica - Estudo e ensino 4. Estatísticas I. Título.

23-168611 CDD-574.192

Índices para catálogo sistemático:

1. Bioquímica : Biologia molecular 574.192

Eliane de Freitas Leite - Bibliotecária - CRB 8/8415

Bioquímica Quantitativa & R

José Maurício Schneedorf Ferreira da Silva

O *software* de programação estatística **R** e sua interface gráfica (*GUI*) **Rstudio** encerram um ambiente de desenvolvimento integrado (*IDE*) de emprego virtualmente ilimitado para diversas áreas do conhecimento. Dentre essas, as *Ciências da Natureza* e, em especial, a *Bioquímica*, *Biologia Molecular*, *Biofísica*, e áreas afins.

São muitas as características que possibilitam o uso do *R & RStudio* para o ensino e aprendizagem em *Bioquímica*. Parte dessas características residem na *Pesquisa Reproduzível* (Gandrud 2018), o qual é resumidamente norteada pela:

1) Disponibilização dos dados originais;
2) Existência de um código para processar esses dados e analisá-los;
3) Documentação dos dados e do código, possibilitando sua reprodutibilidade;
4) Distribuição e acessibilidade do código.

Nesse sentido, a produção de textos, tabelas, gráficos, análise de dados e simulações, podem perfeitamente ajustar-se ao **ensino-aprendizagem** pelo uso de tais princípios e da simultaneidade de texto e códigos, tangenciando um **Ensino Reproduzível**. Tal abordagem permite ao leitor o estudo dos diversos temas tratados por leitura e interpretação, como também de sua apropriação mais convergente, pela execução, modificação, e criação de códigos pertinentes em cada tema. Na presente obra, essas concepções são aplicadas ao *conteúdo quantitativo* e às *relações matemáticas* dos temas abordados em *Bioquímica*.

Essas relações estão presentes nos **livros-texto de Bioquímica** e abrangem, por exemplo, **curvas de titulação de ácidos fracos e aminoácidos**, estudo de **cargas efetivas em biomoléculas**, **características físico-químicas de proteínas e ácidos nucleicos** previstas por análise de sequência, **cinética de enzimas e sua inibição**, **quantidades termodinâmicas e Bioenergética**, **interação ligante-biopolímero**, **estequiometria de reações bioquímicas**, **rotas bioquímicas e redes metabólicas**, dentre outros.

De modo geral, os temas acima são abordados neste material com auxílio do *R & RStudio*. Não obstante, seu conteúdo não possui a pretensão de abordar, para além da superfície, o uso do *R*, do *RStudio*, e mesmo dos temas de *Bioquímica* propostos. Para esses, recomenda-se as fontes tradicionais de tutoriais, livros-texto e internete. E tampouco se arrisca a adentrar no universo da *Bioinformática*, tradicional ou estrutural, como na *Biologia de Sistemas*, *alinhamento de sequências*, *predição estrutural*, *modelagem, dinâmica e atracamento moleculares*, ou nas várias faces dos *estudos ômicos*.

Em resumo, objetiva-se apenas abordar os *conteúdos quantitativos* e *relações matemáticas* presentes em parte da *Bioquímica*, tal como descrito acima, utilizando o *R & Rstudio*. Essa abordagem tangencia a **solução de problemas e simulações por sistema linear de equações, álgebra linear, ajuste linear, não linear, polinomial, linear múltiplo, otimização, minimização, equações diferenciais simples, e análise de sequências**, dentre outros.

Secundariamente, objetiva-se permitir que o leitor possa, a partir da repetição ou modificação de trechos simples de códigos e de *scripts*, reproduzir cálculos, gráficos e/ou tabelas pertinentes aos conteúdos elencados.

Sec 1.1 Conversão de unidades

Quantidades podem ser convertidas com auxílio do R, desde que definidas as unidades envolvidas. Exemplificando, podemos converter e calcular o tempo decorrido de um evento em diferentes unidades, como segue:

```
# Algumas conversões de unidades de tempo
seg <- 1
min <- 60 * seg
hr <- 60 * min
dia <- 24 * hr
anos <- 365 * dia
# Qual o valor em segundos para um dia inteiro ?
dia / seg
```

```
[1] 86400
```

```
# Qual o valor em minutos para um ano inteiro ?
anos / min
```

```
[1] 525600
```

```
# E qual a idade da Terra em segundos (4.5e9 anos) ?
4.5e9 * anos / seg
```

```
[1] 1.41912e+17
```

Também é possível converter unidades entre si, como nas quantidades abaixo:

```
# Conversões de quantidades molares
mmol <- 1 # definições de quantidades
umol <- 1e-3 * mmol # micromol
nmol <- 1e-3 * umol # nanomol
pmol <- 1e-3 * nmol # picomol

# Quantos picomol possui 6,25 mmol ?
6.25 * mmol / pmol
```

```
[1] 6.25e+09
```

Sec 1.2 Versatilidade estrutural em biopolímeros

Biopolímeros, ou biomacromoléculas, podem ser considerados polímeros com unidades monoméricas compostas por biomoléculas. Assim, proteínas, ácidos nucleicos ou glicanos (polissacarídios) são respectivamente formados por aminoácidos (20 tipos codificáveis em proteínas, a partir de 64 códons do código genético), bases nitrogenadas (4 tipos com citosina do DNA substituida por uracila no RNA) e monossacarídios (inferior a 6 tipos).

Do ponto de vista da variabilidade estrutural, tomando-se por base apenas a combinação de monômeros, é possível prever o número de estruturas distintas pela simples relação (Otaki et al. 2005):

$$no.biopolímeros = monômeros^{sequência} \tag{1.1}$$

No 'R' a operação é bem simples, como no exemplo abaixo:

```
# Cálculo de estruturas peptídicas possíveis numa
# sequência de 8 elementos (ex: angiotensina II)

20^8
```

```
[1] 2.56e+10
```

É claro que essa variabilidade simulada não se concretiza na Natureza, posto que o cálculo pressupõe a repetição de qualquer monômero ao longo da sequência, ou de conjuntos ou alterâncias desses. Ou seja, peptídeos com somente um tipo ou dois de aminoácidos, por exemplo, não são fisiologicamente viáveis. Isso se concretiza quando observamos que existem em torno de 35 mil proteínas expressas pelo genoma humano, e cujo tamanho médio encontra-se em torno de 476 resíduos de aminoácidos. Se aplicarmos a equação (1.1) acima para essa situação, encontraríamos...

```
# Cálculo de estruturas proteicas humanas possíveis numa sequência média de
# 476 resíduos de aminoácidos.

20^476
```

```
[1] Inf
```

Ou seja, nem mesmo o `R` é capaz de calcular, uma vez que o tamanho da sequência é limitada computacionalmente no programa a 237 resíduos ($20^{237} = 1.1\text{x}10^{307}$). Ainda que pareça uma limitação, veja que resulta em valor muito acima do número de Avogadro ($6{,}02\text{x}10^{23}$), e mesmo acima do limite computacional para alguns programas matemáticos, tais como os encontrados em algoritmos da internete (`Google`), e programas matemáticos (ex: `Gnu Octave`). Ainda assim, o `Maxima`, programa matemático de distribuição livre, informa que 20^{476} representa um valor com 570 dígitos (algo como 10^{569}).

Em contrapartida, esses cálculos simples também não levam em conta que biomoléculas podem apresentar diversos tipos de *isomeria*, como óptica (D/L), posicional, geométrica (cis/trans), configuracional (syn/anti), ou conformacional (bote/cadeira), o que eleva consideravelmente o número de estruturas possíveis na Natureza.

Sec 2.1 Sistema acetato

A titulação de um ácido fraco é baseada na equação de Henderson-Hasselbach como segue (Po e Senozan 2001) :

$$pH = pKa + log\frac{[A^-]}{[HA]} \tag{2.1}$$

Ocorre que podemos tratar os teores de A^- e HA não em termos absolutos, mas como frações, tanto de base (*fb*), como de ácido (*fa*), tal que:

$$fa + fb = 1 \tag{2.2}$$

Assim, pode-se definir que após uma certa quantidade de base, o valor inicial de HA, em fração unitária, será de *1-fb*; dessa forma, a expressão de Henderson-Hasselbach pode ser escrita como:

$$pH = pKa + log\frac{fb}{1 - fb} \tag{2.3}$$

A partir dessa dedução, pode-se facilmente relacionar que:

$$fb = \frac{10^{(pH-pKa)}}{1 + 10^{(pH-pKa)}} \tag{2.4}$$

E, da mesma forma, pode-se encontrar *fa* como

$$fa = 1 - fb \tag{2.5}$$

Resultando em

$$fa = \frac{1}{1 + 10^{(pH-pKa)}} \tag{2.6}$$

Dessa forma é possível simular pelo *R* uma curva de titulação de um ácido fraco qualquer baseando-se em seu valor de pKa. Com exemplo em meio de acetobactérias, podemos exemplificar o tampão acetato, com valor de pKa de 4,75. Para isso utiliza-se a função `curve` a partir de seus argumentos (`args`), como segue:

```
# Argumentos para uma função
args(curve)
```

```
function (expr, from = NULL, to = NULL, n = 101, add = FALSE,
    type = "l", xname = "x", xlab = xname, ylab = NULL, log = NULL,
    xlim = NULL, ...)
NULL
```

Ou, de forma mais simples:

```
# Curva de titulação para o sistema acetato/ácido acético
pKa = 4.75
curve((1/(1+10^(x-pKa))),0,14)
```

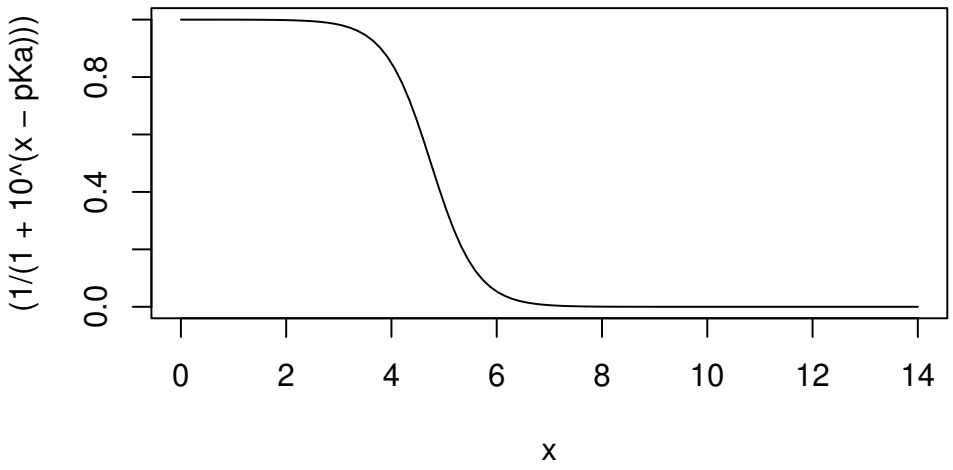


Figura 2.1: Relação entre pH (abscissa) e fração ácida (ordenada) para o par conjugado ácido acético/íon acetato.

Também pode-se fazer o inverso, elaborando um gráfico com a fração *fb*:

```
# Curva de titulação para o sistema acetato/ácido acético
pKa = 4.75
curve(((10^(x-pKa))/(1+10^(x-pKa))),0,14)
```

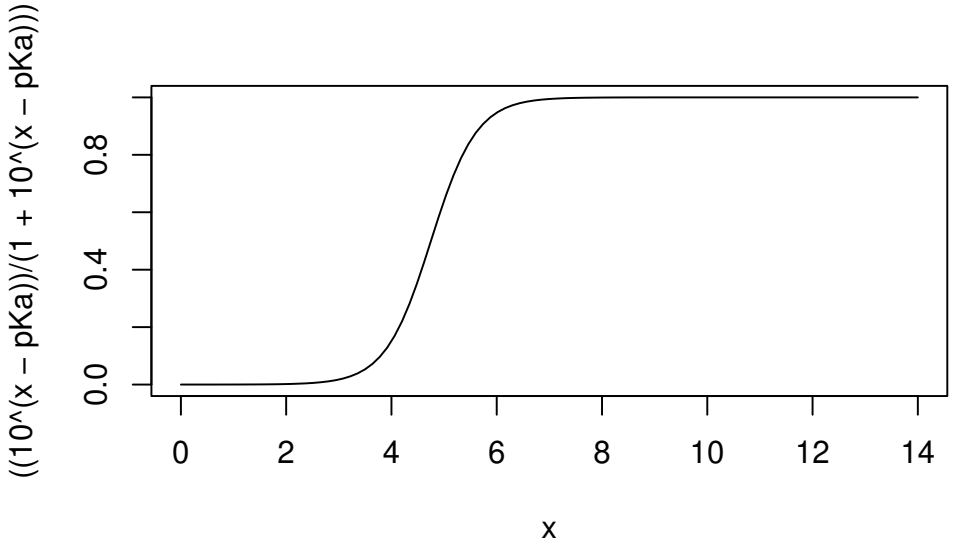


Figura 2.2: Relação entre pH (abscissa) e fração básica (ordenada) para o par conjugado ácido acético/íon acetato.

Sec 2.2 Sistema bicarbonato

Com o procedimento acima pode-se também simular a curva de titulação para o sistema bicarbonato de tamponamento sanguíneo com base nos valores de pKa do par ácido-base conjugado, apenas somando-se as expressões

na equação (2.6), tal que:

$$fa = \frac{1}{1+10^{(pH-pKa1)}} + \frac{1}{1+10^{(pH-pKa2)}} \tag{2.7}$$

Assim,

```
pKa1 = 6.37
pKa2 = 10.20
curve((1/(1+10^(x-pKa1)))+1/(1+10^(x-pKa2)),0,14)
```

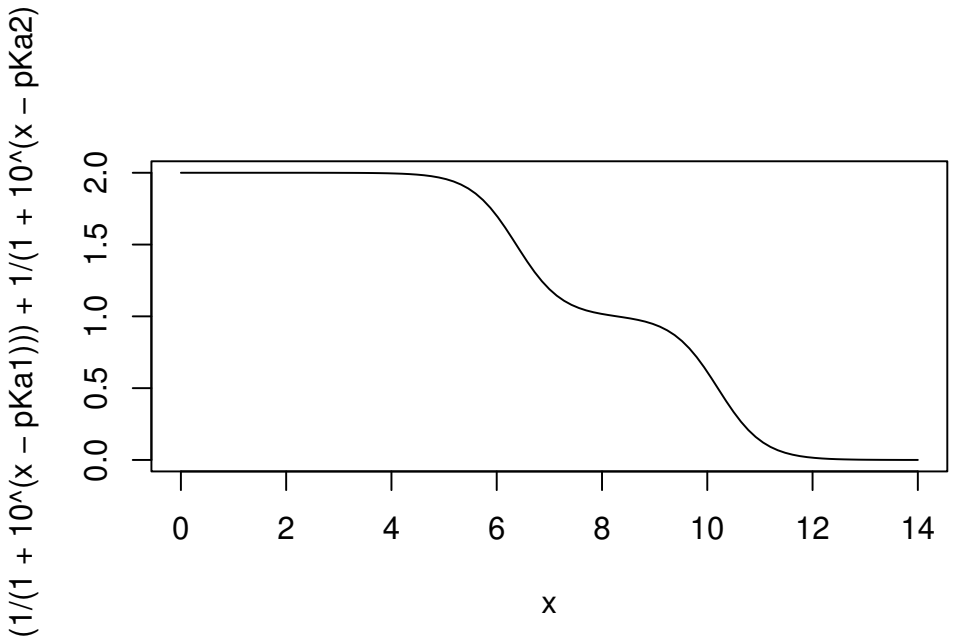


Figura 2.3: Relação entre pH (abscissa) e fração ácida (ordenada) para o par conjugado ácido carbônico/íon bicarbonato.

O gráfico da Figura 2.3 pode ser armazenado em formatos, utilizando-se, por exemplo, o comando `dev.copy`:

```
dev.copy(pdf,"titBicarb.pdf",width=6, height=3) # alternativamente, bmp,
# jpeg, tiff, svg, png
```

E é claro que, partindo-se dos argumentos da função `curve` acima, e da flexibilidade que o pacote interno *Graphics* do R possibilita, pode-se elaborar uma curva mais complexa, como segue:

```
pKa1 = 6.37
pKa2 = 10.20
curve((1/(1+10^(x-pKa1)))+1/(1+10^(x-pKa2)),0,14,
     xlab="pH",ylab="fa",
     main="Titulação de Ácido carbônico, H2CO3/HCO3-",
     type="o", n=50,lwd=2,lty="dotted",
     pch=3,col="blue",cex=1.2) # gráfico de titulação

text(4.7,1.3,"pKa = 6,37") # inserção de texto no gráfico
text(9,0.3,"pKa = 10,20")
abline(0.5,0, lty="dotted") # linha pontilhada em intercepto
# e inclinação específicos
abline(1.5,0, lty="dotted")
```

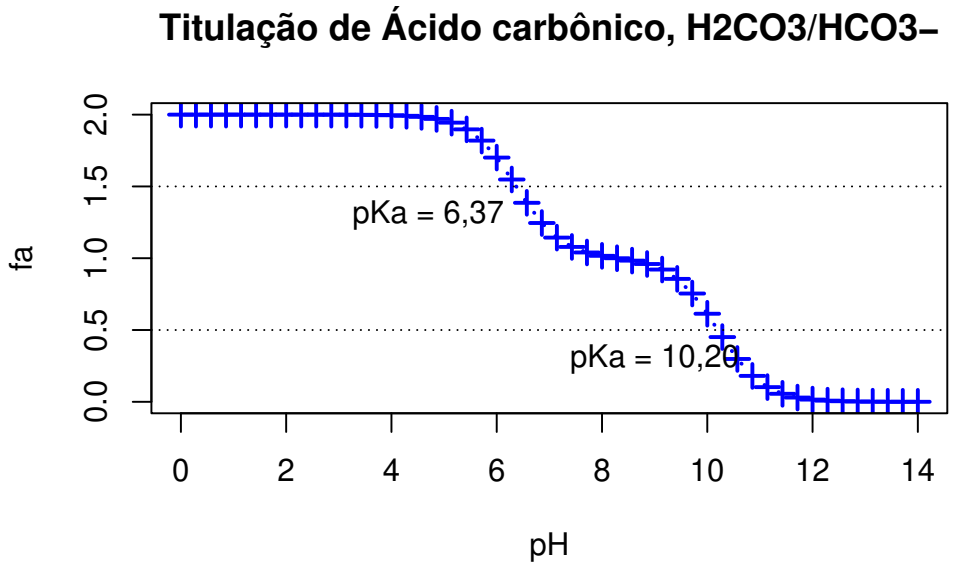


Figura 2.4: Variação de pH com redução da fração ácida em sistema bicarbonato.

A título de ilustração é possível "recuperar" o valor de *pKa* fisiológico acima, ou seja, *pKa1*, utilizando-se o comando `locator()`. Como trata-se de um ponto apenas no gráfico, basta digitar o código `locator(1)` e clicar com o botão esquerdo do mouse no ponto da curva correpondente à fração de 0,5 para *fa*.

```
locator(1) # para mais pontos no gráfico, basta aumentar o valor entre parênteses
```

Observe que à medida em que o valor de *pH* aproxima-se do de *pKa*, a crescente variação em *fa* parece afetar cada vez menos a variação em pH. Isto é a "alma" do sistema tampão, que permite aos organismos resistirem a variações de pH tanto quanto essas estiverem próximas do valor de *pKa* correspondentes (bicarbonato, fosfato, proteínas).

Enquanto o sistema bicarbonato possui dois valores de pKa [1] , um dos quais na faixa de tamponamento fisiológico extracelular, o sistema fosfato que atua intracelularmente possui três valores de pKa, embora também atuando em apenas uma faixa fisiológica.

Sec 2.3 Sistema fosfato

Da mesma forma que simulado para o sistema bicarbonato, podemos elaborar uma curva de titulação para o sistema fosfato de tamponamento, dessa vez considerando seus três valores de pKa correspondentes a cada dissociação do ácido triprótico. Como dantes, a expressão que define a fração *fa* deverá ser tomada como uma soma algébrica, como segue:

$$fa = \frac{1}{1+10^{(pH-pKa1)}} + \frac{1}{1+10^{(pH-pK2)}} + \frac{1}{1+10^{(pH-pKa3)}} \tag{2.8}$$

No *R* isso pode ser feito como abaixo: (2.8)

```
pKa1=2.2
pKa2=7.2
pKa3=12.7

curve((1/(1+10^(x-pKa1)))+
      (1/(1+10^(x-pKa2)))+
```

[1] Obs: o valor de pKa do sistema bicarbonato é de 6,8 quando considerado o CO_2 como fonte de ácido carbônico H_2CO_3 em sua reação com H_2O, como por exemplo, para determinação de parâmetros arteriais em analisador hospitalar (CO_2, HCO_3^-, O_2).

```
      (1/(1+10^(x-pKa3))),
      xlim=c(1,14),
      xlab="pH",ylab="fa",
      main="Titulação de tampão fosfato",
      sub = " As linhas pontilhadas cruzam os valores de pKa"
                  )
abline(v=c(2.2,7.2,12.7),col=c("blue","red","green"),lty="dotted") # adição de
# linhas verticais marcando os valores de pKa
text(1.6,2.5,"pKa1")
text(6.5,1.5,"pKa2")
text(11.8,0.5,"pKa3")
```

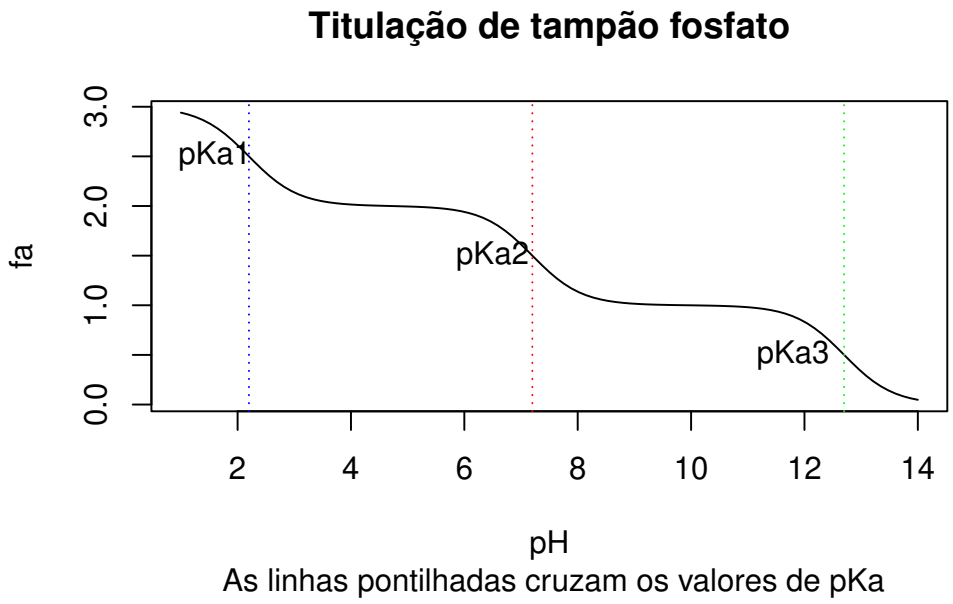


Figura 2.5: Curva de titulação em sistema fosfato de tamponamento.

2.3.1 Titulação de sistemas em geral com programação do R

Como ilustrado no fornecimento de argumentos da função *args*, o 'R' é uma linguagem de programação orientada a objeto, e cujos comandos são estruturados como *funções*. Dessa forma, é possível criar uma função no 'R' para operacionalizar ou automatizar qualquer trabalho computacional.

Uma função pode ser criada basicamente pelas instrução que segue:

```
função.X <- function( arg1, arg2, arg3 )
{
comandos de execução
return( objeto da função )
}
```

Como exemplo, pode-se criar uma função para converter a temperatura de graus Celsius (C) para temperatura absoluta (K), como segue:

```
# Função para conversão de graus Celsius a Kelvin
CtoK <- function (tC) {
  tK <-tC + 273.15
  return(tK)
  }
```

Para executar essa *função CtoK*, basta:

```
# Executando CtoK:
CtoK (37)
```

```
[1] 310.15
```

Tendo isso em mente, também podemos criar uma função que auxilie na elaboração de curvas de titulação, como acima. Essas operações podem ser automatizadas não apenas para o tampão fosfato, mas para qualquer composto sob dissociação em meio aquoso, não importando o número de prótons envolvidos. Para isso, é necessário:

1. Definir uma *função* do *R* que contenha os parâmetros e a operação desejada.
2. Incluir na função uma estrutura de *laço* ou *loop* que permita repetir a operação até exaurido o número de prótons do composto.
3. Definir um vetor do *R* contendo os valores dos pKas do composto.
4. Definir a expressão de *curva* que viabilize a simulação.

Abaixo é apresentado um modelo de código que permite a simulação para o tampão fosfato.

```
#Define função e plot de titulação
fa = function(pH,pKa) {
  x=0
  for(i in 1:length(pKa)) {
    x = x+1/(1 + 10^(pH - pKa[i]))}
  return(x)
}
pKa=c(2.2,7.2,12.7)
curve(fa(x,pKa),1,14, xlab="pH", ylab="fa",
      col=2)
```

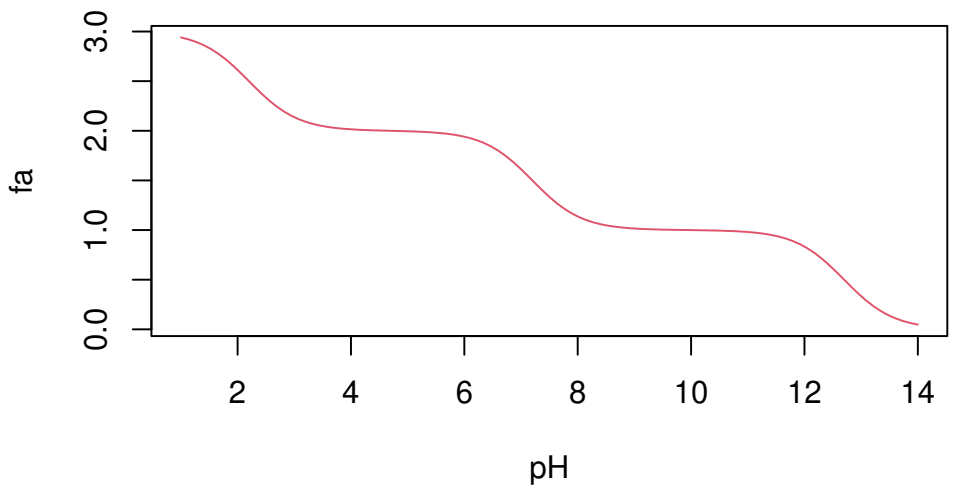


Figura 2.6: Curva de titulação para o tampão fosfato, tal como obtida por recurso de programação no R.

Sec 3.1 Ponto isoelétrico & aminoácidos

De modo geral o ponto isoelétrico, ou *pI*, representa o valor de pH em que uma molécula adquire uma carga líquida nula sob campo elétrico, ou seja, suas cargas positivas anulam-se com as cargas negativas. Normalmente é experimentalmente obtido por medidas cinéticas, tais como *potencial Zeta, eletrofocalização ou eletroforese capilar*. De modo similar, o *ponto isoiônico* refere-se à mesma condição, no entanto na ausência de campo elétrico, podendo ser aferido por *titulação potenciométrica, viscosidade*, ou pela informação estrutural de uma sequência monomérica, tal como ocorre na *sequência primária de proteínas*.

Como todos os 20 aminoácidos que participam da estrutura proteica possuem grupos ionizáveis, tanto em seu esqueleto carbônico como em sua cadeia lateral, é possível prever o ponto isoiônico de um aminoácido em função dos valores de *pKa* apresentados nesses grupos ionizáveis. O *pI* também é denominado comumente por *ponto isoelétrico*, embora essa definição encerre em si uma abrangência teórica mais complexa.

Exemplificando, o ácido glutâmico (Glu, E) apresenta um carboxilato ionizável em sua cadeia lateral, além dos grupos amina (-H_2N) e carboxilato do esqueleto carbônico Figura 3.1:

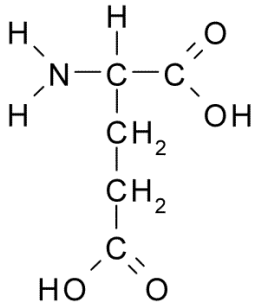


Figura 3.1: Fórmula estrutural planar para o ácido glutâmico (Wikimedia).

Dessa forma, sua *rede de carga líquida*, *qnet*, pode ser determinada a partir da soma da forma ácida (qa) e básica (qb) da molécula, de forma similar como a que foi apresentada a partir da equação (2.8):

$$qnet = qb + qa \tag{3.1}$$

$$qnet = qb + \frac{1}{1 + 10^{pH - pKa}} \tag{3.2}$$

Como trata-se um ácido poliprótico, a Equação 3.2 torna-se:

$$qnet = \sum_{i=1}^{n} (qb + \frac{1}{1 + 10^{pH - pKi}}) \tag{3.3}$$

, com *pKi* como o enésimo valor de pKa. Dessa forma pode-se determinar programaticamente a curva de titulação do ácido glutâmico em função de sua carga, e não da fração ácida. Nessa linha, *qb* representa a forma do composto em base, o que para Glu apresentará os valores de *-1* para os dois carboxilatos, e de *0* para o grupo amina, sendo necessário compor um vetor adicional para *qb*.

3

```
# Titulação de Glu

qNet <- function(pH, qB, pKa) {
  x <- 0
  for (i in 1:length(qB)) {
    x <- x + qB[i] + 1 / (1 + 10^(pH - pKa[i]))
  }
  return(x)
}
qB <- c(-1, 0, -1)
pKa <- c(2.2, 9.7, 4.3)

curve(qNet(x, qB, pKa), 1, 12, xlab = "pH", ylab = "qNet")

abline(0, 0, lty = "dotted")
```

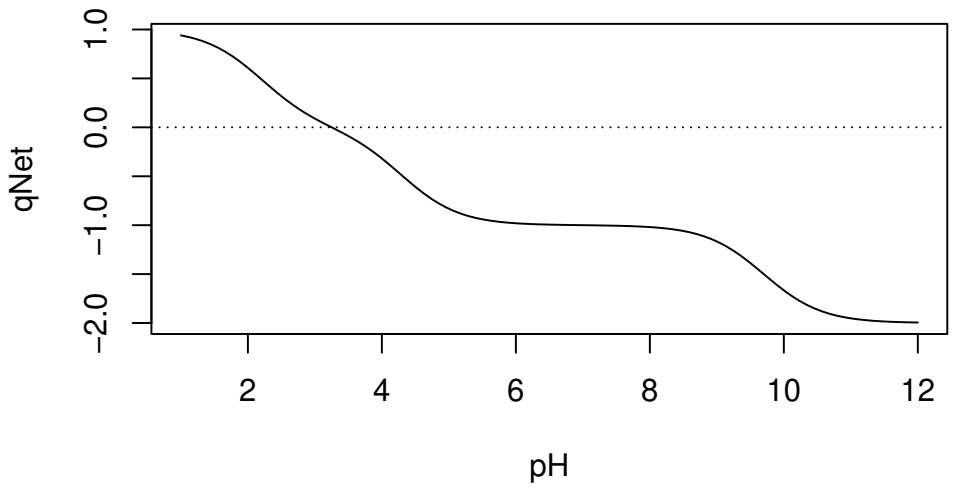


Figura 3.2: Curva de titulação para o ácido glutâmico. A curva intercepta a linha pontilhada no ponto isoiônico do Glu.

Manualmente é possível identificar o valor de pI para o ácido glutâmico por uma função do R, tal como `locator()` visto anteriormente. Mas também é possível acessar esse valor automaticamente, aplicando um comando que encontre a raíz dessa função, ou seja, o valor de pH que corresponda a um valor nulo para *qnet*. Para isto, exemplifica-se o uso de **uniroot**, no qual define-se a função matemática pretendida, bem como os limites inferior e superior para a busca pelo algoritmo, como segue:

```
# Cálculo de pI
f <- function(pH) {
  qNet(pH, qB, pKa)
}
str(uniroot(f, c(2, 5)))
```

```
List of 5
 $ root      : num 3.25
 $ f.root    : num -4.8e-06
 $ iter      : int 4
 $ init.it   : int NA
 $ estim.prec: num 6.1e-05
```

Esse resultado traduz-se como um pI de 3,25 (`root`), em 4 iterações, com uma estimativa de precisão de 6,1x10^{-5}, e erro associado de -4,8x10^{-6}.

Essa forma de se obter um valor empregando-se o cálculo numérico é por vezes denominada **solução numérica**. Por outro lado, pode-se obter o valor de pI para o Glu por um procedimento mais simples, normalmente encontrado nos livros-texto sobre o assunto, e que assume a forma abaixo:

$$pI = \frac{pKa1 + pKa2}{2} \tag{3.4}$$

No nosso exemplo, o pI envolverá os pKas dos dois carboxilatos, o que resultará em *(2,3+4,2)/2*, ou seja, *3,25* ! Nada mal para uma aproximação, não ? Esse procedimento envolvendo a solução de um problema matemático a partir de parâmetros do sistema é denominado método ou **solução analítica**. Essa solução também pode ser exemplificada pelo parâmetro obtido em função da observação do comportamento gráfico da titulação, como nas figuras acima.

Agora, pra que nos serve um procedimento numérico mais complexo, se uma simples equação analítica já nos resolve o problema de se encontrar o valor de pI para o ácido glutâmico ? Bom, extamente pra isso, para solução de problemas mais complexos. Um pouco menos retórico, entretanto, pode-se afirmar que a *solução numérica* funciona melhor para sistemas onde a *solução analítica* por vezes não é suficiente ou torna-se mesmo impossível, como na solução de equações com dezenas de parâmetros.

Sec 3.2 Ponto isoiônico & biopolímeros

Uma situção nesse tema pode ser ilustrada pela obtenção do valor de pI para uma proteína. Exemplificando, a lisozima humana, enzima de estrutura terciária composta por 130 resíduos de aminoácidos. Nesse caso, a *solução analítica* esbarra na complexidade em se identificar quais desses resíduos são ionizáveis em solução aquosa, e quais estariam envolvidos numa distribuição que resultasse numa carga líquida nula para a molécula.

Para esse sistema mais complexo é necessário ampliar um pouco a função definida para o ácido glutâmico, computando-se no vetor de *qb* as cargas em base dos 7 aminoácidos com cadeias laterais ionizáveis, e atribuir um novo vetor para o quantitativo de cada resíduo ionizável presente na lisozima. O código abaixo exemplifica essa solução, calcula o pI da enzima, e elabora o gráfico de sua titulação, embora essa ordem não seja relevante, posto que o pI é calculado numericamente, e não graficamente.

```
# Titulação de Lisozima e Determinação de pI

# Define função para qNet
qNet <- function(pH, qB, pKa, n) {
  x <- 0
  for (i in 1:length(qB)) {
    x <- x + n[i] * qB[i] + n[i] / (1 + 10^(pH - pKa[i]))
  }
  return(x)
}

# Define pKas de aCOOH, aNH3 e as 7 cadeias laterais de AA
pKa <- c(2.2, 9.6, 3.9, 4.1, 6.0, 8.5, 10.1, 10.8, 12.5)

# Define qB, as cargas de cada aminoácido na forma básica
qB <- c(-1, 0, -1, -1, 0, -1, -1, 0, 0)

ionizavel <- c(
  "aCOOH", "aNH3", "Asp", "Glu", "His", "Cys", "Tyr",
  "Lys", "Arg"
)
n <- c(1, 1, 7, 3, 1, 8, 6, 5, 14) # Lista para quantidades de resíduos
# ionizáveis na lisozima (cada elemento representa a quantidade
# de aCOOH, aNH3, e determinado AA na enzima)

# Cálculo de pI
```

```
f <- function(pH) {
  qNet(pH, qB, pKa, n)
}
str(uniroot(f, c(1, 13))) # estimativa de pI entre 10 e 12
```

```
List of 5
 $ root      : num 9.46
 $ f.root    : num 3.3e-07
 $ iter      : int 7
 $ init.it   : int NA
 $ estim.prec: num 6.1e-05
```

```
# Gráfico de titulação
curve(qNet(x, qB, pKa, n), 1, 12, xlab = "pH", ylab = "qNet")
abline(0, 0, lty = 3)
```

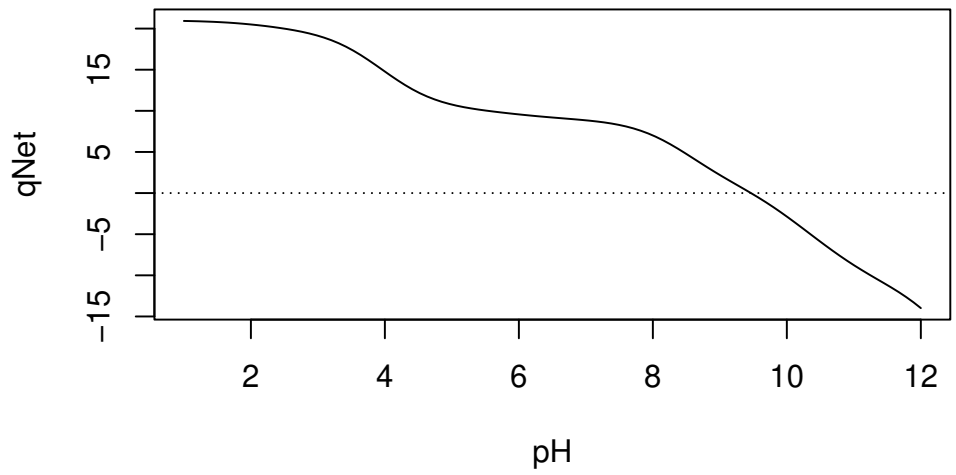


Figura 3.3: Curva de titulação para lisozima.

Observe que o valor encontrado para pI da lisozima foi de 9,46; ou seja, em pH 9,46 a enzima apresenta rede de carga líquida nula, como também pode ser verificado na representação gráfica.

Sec 3.3 Ponto isoiônico & bibliotecas do R

Não obstante a precisão do cálculo de pI pela *solução numérica* realizada para a lisozima, uma das características mais fascinantes do programa reside no uso de bibliotecas (`packages`), não sendo diferente para determinção de propriedades de biopolímeros, tais como pI.

Entre as bibliotecas existentes para propriedades físico-químicas de proteínas e ácidos nucleicos exemplifica-se o pacote `seqinr`, *Biological Sequences Retrieval and Analysis* [1], de análise e visualização exploratória de biopolímeros. Para uso desse pacote, contudo, faz-se necessário a obtenção da sequência primária da proteína, representada em código de uma letra. Pode-se obter a sequência primária da lisozima pelo sítio do *National Center for Biotechnology Information*, NCBI [2]. Um truque rápido envolve:

1) digitar o nome da proteína;

2) selecionar entre as opções resultantes;

[1] Pacote seqinr: https://cran.r-project.org/web/packages/seqinr/index.html
[2] NCBI. https://www.ncbi.nlm.nih.gov/protein

3) clicar em FASTA para obter a sequência primária de 1 letra.

4) copiar a sequência da proteína apresentada para o **seqinr**.

Presupondo-se que a biblioteca **seqinr** esteja instalada, e que a sequência tenha sido obtida para a lisozima (busca por *CAA32175* ou *lysozyme [Homo sapiens]*), pode-se encontrar o valor do pI para a mesma pelo código que segue:

```
library(seqinr)
lisozima <- s2c("KVFERCELARTLKRLGMDGYRGISLANWMCLAKWESGYNTRATNYNAGDR
                 STDYGIFQINSRYWCNDGKTPGAVNACHLSCSALLQDNIADAVACAKRVV
                 RDPQGIRAWVAWRNRCQNRDVRQYVQGCGV")
# converte sequência de string em vetor de caracteres
computePI(lisozima)
```

```
[1] 9.2778
```

Veja que o valor de pI pelo pacote, 9,28, foi bem próximo do encontrado pela *solução numérica* acima. Isto deve-se ao uso de algoritmos distintos para ambos, bem como o cômputo dos valores de pKa distintos para o **seqinr**. Exemplificando essa variação, o próprio **seqinr** apresenta valores de pKa diferentes, em função da base de dados buscada. Para verificar isso, digite o comando abaixo e visualize e variável *pK* resultante.

```
library(seqinr)
data(pK)
```

Complementarmente, pode-se também comparar o valor de pI da lisozima com o algoritmo utilizado pelo banco de dados no sítio [3]. Para isto, basta colar a sequência de resíduos no campo disponível e clicar o cômputo de pI. Veja que o valor resultante de 9,28 coincide com o do algoritmo utilizado pelo pacote **seqinr** do R.

```
library(knitr)
knitr::kable(pK, "pipe", caption = "Tabela de valores de pKa para aminoácidos
             a partir de diversas fontes, extraída do pacote seqinr.")
```

Tabela 3.1: Tabela de valores de pKa para aminoácidos a partir de diversas fontes, extraída do pacote seqinr.

	Bjellqvist	EMBOSS	Murray	Sillero	Solomon	Stryer
C	9.00	8.5	8.33	9.0	8.3	8.5
D	4.05	3.9	3.68	4.0	3.9	4.4
E	4.45	4.1	4.25	4.5	4.3	4.4
H	5.98	6.5	6.00	6.4	6.0	6.5
K	10.00	10.8	11.50	10.4	10.5	10.0
R	12.00	12.5	11.50	12.0	12.5	12.0
Y	10.00	10.1	10.07	10.0	10.1	10.0

Existem outros pacotes do R que analisam sequências de aminoácidos e nucleotídeos, incluindo o cálculo de pI, entre os quais vale mencionar o Peptides [4].

[3] Expasy. https://web.expasy.org/compute_pi/

[4] pacote Peptides. https://cran.r-project.org/web/packages/Peptides/index.html

Sec 4.1 Composição de aminoácidos

Proteínas constituem biopolímeros formados por 20 aminoácidos. Dessa forma pode-se avaliar facilmente a composição de qualquer proteína disponível em banco de dados, como o *PDB* visto no capítulo Capítulo 3. Tomando-se a albumina de soro humano como exemplo, código *AAA98797* do *National Center for Biotechnology Information*, é possível contabilizar os 20 tipos de aminoácidos que compõe sua sequência. Para isto basta considerar a sequência *FASTA* como uma *string*, e extrair a quantidade de cada letra, utilizando-se a biblioteca *stringr*, como segue.

Primeiro, obtém-se a sequência *FASTA* da albumina de soro.

```
seq <- "MKWVTFISLLFLFSSAYSRGVFRRDAHKSEVAHRFKDLGEENFKALVLIAFAQYLQQCPFEDHVKLVNEV
TEFAKTCVADESAENCDKSLHTLFGDKLCTVATLRETYGEMADCCAKQEPERNECFLQHKDDNPNLPRLV
RPEVDVMCTAFHDNEETFLKKYLYEIARRHPYFYAPELLFFAKRYKAAFTECCQAADKAACLLPKLDELR
DEGKASSAKQRLKCASLQKFGERAFKAWAVARLSQRFPKAEFAEVSKLVTDLTKVHTECCHGDLLECADD
RADLAKYICENQDSISSKLKECCEKPLLEKSHCIAEVENDEMPADLPSLAADFVESKDVCKNYAEAKDVF
LGMFLYEYARRHPDYSVVLLLRLAKTYETTLEKCCAAADPHECYAKVFDEFKPLVEEPQNLIKQNCELFE
QLGEYKFQNALLVRYTKKVPQVSTPTLVEVSRNLGKVGSKCCKHPEAKRMPCAEDYLSVVLNQLCVLHEK
TPVSDRVTKCCTESLVNRRPCFSALEVDETYVPKEFNAETFTFHADICTLSEKERQIKKQTALVELVKHK
PKATKEQLKAVMDDFAAFVEKCCKADDKETCFAEEGKKLVAASQAALGL"
```

Pode-se observar que há espaços vazios, que podem ser omitidos por razões estéticas ou não, já que o pacote *stringr* não os contabilizará, contrariamente a pacotes mais específicos para sequências biológicas, como o *seqinr* visto anteriormente. Mas se desejar omitir esses espaços, basta executar o código abaixo.

```
seq <- seq[seq != "\n"]
seq # operação booleana != significa "não"
```

A seguir, obtém-se o quantitativo de uma letra específica da sequência.

```
library(stringr)
aa <- str_count(seq, pattern = "A")
aa
```

```
[1] 63
```

Veja que o comando `str_count` contabiliza apenas a letra "A" na sequência. Dessa forma, é possível obter todos os 20 aminoácidos, repetindo-se esse comando.

```
library(stringr)
ala <- str_count(seq, pattern = "A")
arg <- str_count(seq, pattern = "R")
asn <- str_count(seq, pattern = "N")
asp <- str_count(seq, pattern = "D")
cys <- str_count(seq, pattern = "C")
glu <- str_count(seq, pattern = "E")
gln <- str_count(seq, pattern = "Q")
gly <- str_count(seq, pattern = "G")
his <- str_count(seq, pattern = "H")
ile <- str_count(seq, pattern = "I")
leu <- str_count(seq, pattern = "L")
lys <- str_count(seq, pattern = "K")
```

```
met <- str_count(seq, pattern = "M")
phe <- str_count(seq, pattern = "F")
pro <- str_count(seq, pattern = "P")
ser <- str_count(seq, pattern = "S")
thr <- str_count(seq, pattern = "T")
trp <- str_count(seq, pattern = "W")
tyr <- str_count(seq, pattern = "Y")
val <- str_count(seq, pattern = "V")
```

E, para visualizar o resultado numa tabela:

```
aa_3abrev <- c("Ala", "Arg", "Asn", "Asp", "Cys", "Glu",
    "Gln", "Gly", "His", "Ile", "Leu", "Lys", "Met",
    "Phe", "Pro", "Ser", "Thr", "Trp", "Tyr", "Val")
aa_quant <- c(ala, arg, asn, asp, cys, glu, gln, gly,
    his, ile, leu, lys, met, phe, pro, ser, thr, trp,
    tyr, val)  # vetor com o quantitativo de aminoácidos da proteína
aa_seq <- data.frame(aa_3abrev, aa_quant)  # dataframe com os resultados
colnames(aa_seq) <- c("Tipo", "Qtde")  # renomear as colunas


# Composição de aminoácidos em albumina de soro
# humano
aa_seq  # apresenta a tabela
```

```
   Tipo Qtde
1   Ala   63
2   Arg   27
3   Asn   17
4   Asp   36
5   Cys   35
6   Glu   62
7   Gln   20
8   Gly   13
9   His   16
10  Ile    9
11  Leu   64
12  Lys   60
13  Met    7
14  Phe   35
15  Pro   24
16  Ser   28
17  Thr   29
18  Trp    2
19  Tyr   19
20  Val   43
```

O 'R' possui alguns comandos para a geração estética de tabelas, entre os quais os incluídos no pacote 'knitr', como segue:

```
library(knitr) # para gerar a tabela

knitr::kable(aa_seq, caption = "Composição de aminoácidos em albumina
             # de soro humano.", "pipe") # tabela
```

Tabela 4.1: Composição de aminoácidos em albumina # de soro humano.

Tipo	Qtde
Ala	63
Arg	27

Tipo	Qtde
Asn	17
Asp	36
Cys	35
Glu	62
Gln	20
Gly	13
His	16
Ile	9
Leu	64
Lys	60
Met	7
Phe	35
Pro	24
Ser	28
Thr	29
Trp	2
Tyr	19
Val	43

Observe que, à despeito do resultado obtido, houve certo trabalho em se obter a composição da albumina, a partir das 20 linhas modificadas para cada aminoácido. Uma alternativa mais prática consiste em considerar um *loop* que execute a extração da informação desejada para um vetor contendo a abreviação de uma letra para cada aminoácido.

```
aa_1abrev <- c("A", "R", "N", "D", "C", "E", "Q", "G", "H", "I", "L", "K", "M",
               "F", "P", "S", "T", "W", "Y", "V")
for (i in aa_1abrev) {
  aa_quant2 <- str_count(seq, pattern = aa_1abrev)
  return(aa_quant2) # sintaxe opcional para função com apenas uma saída
}


aa_seq <- data.frame(aa_3abrev, aa_quant2) # dataframe com os resultados
colnames(aa_seq) <- c("Tipo", "Qtde") # renomear as colunas
knitr::kable(aa_seq, caption = "Composição de aminoácidos em albumina de soro
             humano (uso de loop).", "pipe") # tabela
```

Tabela 4.2: Composição de aminoácidos em albumina de soro humano (uso de loop).

Tipo	Qtde
Ala	63
Arg	27
Asn	17
Asp	36
Cys	35
Glu	62
Gln	20
Gly	13
His	16
Ile	9
Leu	64
Lys	60
Met	7
Phe	35
Pro	24
Ser	28
Thr	29
Trp	2

Tipo	Qtde
Tyr	19
Val	43

Dessa forma obtém-se o mesmo resultado, mas com menor consumo de memória e maior velocidade de processamento, características em qualquer lógica de programação. Apesar do *loop* exemplificar uma automação, a função `str_count` retém em si um *loop* interno, já que aplica uma função de contagem de elementos a uma sequência, a partir de um padrão pré-definido (o vetor *aa_1abrev*, no caso). Dessa forma, pode-se simplificar ainda mais o script, não necessitando do *loop* externo.

```
str_count(seq, pattern = aa_1abrev)
```

```
 [1] 63 27 17 36 35 62 20 13 16  9 64 60  7 35 24 28 29  2 19 43
```

Diversas funções do R exibem esse *loop* interno, e que pode ser aplicado em vetores, listas, matrizes e *dataframes* (planilha de dados). As mais simples envolvem a aplicação de uma função pré-programada do R a um vetor, por ex:

```
y <- c(1, 2, 4, 8, 16, 32)
mean(y)
```

```
[1] 10.5
```

```
sum(y)
```

```
[1] 63
```

Outra vetorização frequente decorre da aplicação de uma *função* de usuário a vetor, reduzindo a necessidade de repetição de comandos, como no exemplo abaixo:

```
# Tamanho médio estimado de uma proteína a partir do no. de resíduos de
# aminoácidos
prot.tamanho <- function(x) {
  MM <- x * 110 # 'x' representa o número de aminoácidos da proteína
  return(MM)
}

prot.tamanho(575) # no. de resíduos de aminoácidos de albumina humana
```

```
[1] 63250
```

Outra forma de vetorização envolve a família de funções `apply`, composta pelos comandos `apply`, `sapply`, `tapply`, `lapply`, e `mapply`. Embora possuam processamento mais rápido que funções de *loop* externo para uso de matrizes muito complexas, cada qual é voltado a um objeto distinto ou situação específica do R (retorno de lista, vetor ou matriz), permite o uso de `subset` (subconjuntos de dados), utiliza funções do R ou funções previamente definidas pelo usuário, e roda em apenas uma linha de comando. Essas vantagens contrapõe-se ao uso de *loop for* aplicado para vetores. Contudo, a vetorização opera muito bem quando se deseja aplicar ou mapear uma função a um vetor/matriz/lista. Quando, por outro lado, se deseja aplicar uma função cujo resultado dependa de mais de um vetor/matriz/lista, o *loop for* torna-se indispensável, como na titulação de ácidos fracos do capítulo Capítulo 3.

Seja qual for o método empregado (e aí vale destacar as chamadas *boas práticas de programação* na construção de *scripts* [1]), é possível construir uma composição mais geral para o conjunto de resíduos da proteína. Assim, podemos obter qualquer relação quantitativa a partir da sequência, já que a estamos tratando como uma *string*. Exemplificando, uma tabela contendo a classe de cada aminoácido que compõe a sequência.

```
aa_ac <- aa_seq[4, 2] + aa_seq[6, 2] # AA ácicos
aa_bas <- aa_seq[2, 2] + aa_seq[9, 2] + aa_seq[12, 2] # AA básicos
aa_arom <- aa_seq[14, 2] + aa_seq[18, 2] + aa_seq[19, 2] # AA aromáticos
aa_alif <- aa_seq[10, 2] + aa_seq[11, 2] + aa_seq[15, 2] + aa_seq[1, 2] +
```

[1] Algumas práticas de programação (*Best Codes*): 1) organizar um projeto em pastas (ex: dados,figuras,scripts) ou criar um pacote do R como opção; 2) criar seções num código pra facilitar localização; 3) nomear os *code chunks* (pedaços de código); 4) colocar no início do código as bibliotecas utilizadas, fontes, e chamada de dados (evita procurar algo necessário pro *script* rodar ao longo do código); 5) identar, preferivelmente com 1 ou 2 comandos por linha; 6) parâmetros de função sempre dentro de função; 7) evitar parâmetros globais; 8) não usar 'attach'; 8) usar parâmetros com nomes intuitivos (e não x e y; ex: nome_função); 9) atribuir nomes à objetos com uma das três convenções nominais (ex: KiCompet, ki_compet, ki.compet).

```
  aa_seq[20, 2] # AA alifáticos
aa_pol <- aa_seq[3, 2] + aa_seq[5, 2] + aa_seq[7, 2] + aa_seq[8, 2] +
  aa_seq[13, 2] + aa_seq[16, 2] + aa_seq[17, 2] ## AA polares neutros
```

Agora, ao invés de se construir uma tabela com a contagem desses grupos, façamos o percentual dos mesmos, para uma visão mais geral da sequência.

```
aa_tot <- str_count(seq, pattern = "") # comprimento da sequência
class_perc <- round(c(aa_ac, aa_bas, aa_arom, aa_alif, aa_pol) / aa_tot * 100)
```

E agora, sim, constroi-se a tabela.

```
aa_class <- c("ácido", "básico", "aromático", "alifático", "polar")
aa_perc <- data.frame(aa_class, class_perc) # dataframe com os resultados
colnames(aa_perc) <- c("Classe", "%") # renomear as colunas
knitr::kable(aa_perc, caption = "Distribuição de classes de aminoácidos
             em albumina humana.", "pipe") # tabela
```

Tabela 4.3: Distribuição de classes de aminoácidos em albumina humana.

Classe	%
ácido	16
básico	17
aromático	9
alifático	33
polar	24

Percebe-se pela tabela acima que a distribuição de classes de resíduos de aminoácidos na albumina é razoavelmente homogênea, o que contribui para sua função anfotérica de transporte para compostos polares (ex:cálcio) e apolares (ex: colesterol, ácidos graxos).

Sec 4.2 Tabela de Purificação de Proteínas & R como planilha eletrônica

Não obstante a facilidade com que podemos elaborar/editar planilhas eletrônicas convencionais (*spreadsheet*; ex: MS Excel, Libreoffice Calc, Gnumeric, etc), o 'R' também permite trabalhar-se com planilhas. Ainda que menos intuitivo como as mencionadas, a scriptagem no R permite a elaboração/edição de planilhas de alta complexidade, dada a natureza da programação estatística que envolve a suite.

Para exemplificar a construção de uma planilha simples, tomemos como exemplo uma *Tabela de Purificação de Proteínas*, usualmente utilizada em Biotecnologia e áreas afins. A forma mais simples de construção de uma planilha envolve 1) a elaboração individual de vetores, e 2) a união dos vetores em uma planilha.

Os procedimentos para purificação (ou isolamento, fracionamento) proteica envolvem técnicas como **tratamento químico (precipitação por sulfato de amônio, acetona)**, **tratamento ácido**, **tratamento térmico**, **diálise**, **cromatografia (filtração molecular, troca-iônica, afinidade, fase reversa)**, entre outros. Para aferição do grau de pureza da amostra obtida utilizam-se normalmente a **eletroforese** simples, **focalização isoelétrica**, **eletroforese 2D**, uso de **anticorpos monoclonais**, e **ensaios de atividade** específicos, dentre vários.

Para a tabela de purificação são exigidos somente os vetores de **massa de amostra** e de **atividade enzimática da amostra**, obtidos em cada etapa de purificação. Uma planilha simples poderia ser construida como:

```
# Elaboração de planilha simples de purificação de enzima
# (cada elemento do vetor representa uma etapa de purificação)

# 1. Definição dos vetores principais:
prot.total <- c(6344, 302, 145, 34, 10, 3.8) # proteína, mg
ativ.tot <- c(200, 122, 106, 70, 53, 24) * 1000 # atividade, U

# 2. Construção da planilha:
purif.plan <- data.frame(prot.total, ativ.tot)
purif.plan
```

```
  prot.total ativ.tot
1     6344.0   200000
2      302.0   122000
3      145.0   106000
4       34.0    70000
5       10.0    53000
6        3.8    24000
```

A planilha construida compõe agora um 'dataset' do R. Há outras formas de construção simples, também, como o uso da função 'cbind' (pra união de colunas) ou 'rbind' (união de linhas; rows):

```
purif.plan2 <- cbind(prot.total, ativ.tot)
purif.plan2
```

```
     prot.total ativ.tot
[1,]     6344.0   200000
[2,]      302.0   122000
[3,]      145.0   106000
[4,]       34.0    70000
[5,]       10.0    53000
[6,]        3.8    24000
```

Seja qual for o procedimento, pode-se alterar os nomes das colunas, como segue:

```
# Edição de nome de colunas
colnames(purif.plan2) <- c("totalProt", "enzAtiv")
purif.plan2
```

```
     totalProt enzAtiv
[1,]    6344.0  200000
[2,]     302.0  122000
[3,]     145.0  106000
[4,]      34.0   70000
[5,]      10.0   53000
[6,]       3.8   24000
```

Como numa planilha convencional, também é possível se criar novos vetores calculados a partir dos iniciais:

```
purif.plan3 <- data.frame(prot.total, ativ.tot, ativ.tot / prot.total)
options(digits = 1) # opção para no. de casas decimais
colnames(purif.plan3) <- c("prot.total", "ativ.tot", "ativ.specif")
rownames(purif.plan3) <- c("extr.bruto", "NH4SO2", "acetona",
                           "Sephadex G-100", "DEAE-celulose", "C8-fase rev")
purif.plan3
```

```
               prot.total ativ.tot ativ.specif
extr.bruto           6344    2e+05          32
NH4SO2                302    1e+05         404
acetona               145    1e+05         731
Sephadex G-100         34    7e+04        2059
DEAE-celulose          10    5e+04        5300
C8-fase rev             4    2e+04        6316
```

A planilha pode ser editada em seus valores, também, bastando pra isso atribuir um novo nome para que as modificações sejam salvas:

```
# Edição simples de planilha (alterações de valores e nomes de colunas)

purif.plan4 <- edit(purif.plan3) # ou data.entry( )
```

É claro, também, que se pode importar os dados de uma planilha já construida em outro programa. Exemplificando para uma planilha salva como *CSV*:

```
# Importação de dados de outra planilha (CSV):
```

```
# 1. Importação com nome da planilha desejada:
purif.plan5 <- read.table("planilha.csv", header = T, sep = ",")

# 2. Importação com tela de busca da planilha desejada:
purif.plan5 <- frame <- read.csv(file.choose())
```

Um grande número de operações pode ser conduzido em planilhas no 'R', tais como inserção, deleção, modificação, agregação (*merge*), filtragem, extração de subconjunto, operações matemáticas e cálculos estatísticos (média, desvio-padrão, etc). Pode-se também converter a planilha em uma tabela de visual mais completo, por uso da biblioteca 'tibble', parte de um conjunto de pacotes utilizado em ciência de dados denominado *Tidyverse* :

```
library(tibble)
purif.plan6 <- as_tibble(purif.plan3)
purif.plan6
```

```
# A tibble: 6 x 3
  prot.total ativ.tot ativ.specif
       <dbl>    <dbl>       <dbl>
1     6344     200000        31.5
2      302     122000       404.
3      145     106000       731.
4       34      70000      2059.
5       10      53000      5300
6        3.8    24000      6316.
```

O *Tidyverse* compõe um ecossistema de pacotes do 'R' que comungam da mesma filosofia, gramática e estrutura de dados. Entre esses pacotes inlcui-se o 'tibble' (tabelas), 'ggplot2' (gráficos de alta qualidade visual), e 'dplyr' (manipulação de dados). No que tange ao 'dplyr', é bastante flexível a criação e edição de planilhas, e que são convertidas a tabelas, tal como segue para o exemplo da purificação acima:

```
# Tabela de purificação de enzima com pacote 'dplyr':

library(dplyr)
purif.plan7 <- mutate(purif.plan, ativ.esp = ativ.tot / prot.total)
purif.plan7
```

```
  prot.total ativ.tot ativ.esp
1       6344    2e+05       32
2        302    1e+05      404
3        145    1e+05      731
4         34    7e+04     2059
5         10    5e+04     5300
6          4    2e+04     6316
```

Observe que com o pacote 'dplyr' a inserção de uma nova coluna não requereu um novo vetor para nomes de colunas ('colnames'). Além disso, a planilha final foi elaborada junto à gramática do pacote 'tibble' do *Tidyverse*. Essa facilidade também se extende para os processos de edição e filtragem da planilha, tal a extração de uma coluna modificada:

```
ativ.tot.kU <- transmute(purif.plan7, ativ.tot = ativ.tot / 1e3)
ativ.tot.kU # vetor de atividade específica em U x 10^3
```

```
  ativ.tot
1      200
2      122
3      106
4       70
5       53
6       24
```

Dessa forma pode-se construir uma tabela completa de purificação, elencando-se, além da *atividade específica*, o *nível de purificação* (de quantas vezes a atividade específica aumentou em relação à da amostra inicial) e o *rendimento* obtido (de quantas vezes o teor da enzima alvo reduziu em relação à amostra inicial - atividade remanescente):

```
purif.plan8 <- mutate(purif.plan7,
  purif = ativ.esp / ativ.esp[1], # nível de purificação
  rend.perc = 100 * ativ.tot / ativ.tot[1]
) # rendimento percentual

# Convetendo à tabela...
library(knitr)
knitr::kable(purif.plan8, caption = "Tabela de purificação para uma enzima", "pipe")
```

Tabela 4.4: Tabela de purificação para uma enzima

prot.total	ativ.tot	ativ.esp	purif	rend.perc
6344	2e+05	32	1	100
302	1e+05	404	13	61
145	1e+05	731	23	53
34	7e+04	2059	65	35
10	5e+04	5300	168	26
4	2e+04	6316	200	12

Adicionalmente, o 'R' possui alguns pacotes que agilizam a criação/edição de planilhas de modo interativo e mais próximo ao de uma planilha eletrônica convencional, dentre os quais destaca-se o pacote 'DT', uma biblioteca elaborada em *JavaScript* que produz uma planilha editável em *HTML*:

```
library(DT)
purif.plan9 <- as.data.frame(purif.plan8)
rownames(purif.plan9) <- c("extr.bruto", "NH4SO2", "acetona",
                           "Sephadex G-100", "DEAE-celulose",
                           "C8-fase rev") # converte a tabela de purificação
# em planilha para se utilizada pelo pacote DT
datatable(purif.plan9) %>% formatRound(1:5, 1) # colunas com 1 casa decimal
```

Por tratar-se de saída em HTML interativo, o resultado do trecho de código precisa ser omitido para a correta compilação pelo 'R' (eval=FALSE, include=TRUE).

A biblioteca 'DT' permite, entre outros, reordenamento, filtragem, e mesmo edição dos valores, de modo interativo (basta clicar na célula desejada):

```
DT::datatable(purif.plan9, editable = "cell")
```

Sec 4.3 Interação de oxigênio com mioglobina e hemoglobina

Tanto a mioglobina (*PD 1MBO*) como a hemoglobina humanas (*PDB 6BB5*) constituem proteínas de transporte do oxigênio molecular. A hemoglobina, de estrutura quaternária, o faz dos pulmões aos tecidos, enquanto que a mioglobina, terciária, o distribui entre esses. Suas curvas de saturação com oxigênio são bem conhecidas em livros-texto, cujo aprendizado pode agregar valor quando simuladas.

Dessa forma, pode-se considerar a ligação do O_2 à mioglobina como uma fração de saturação y dada em função de sua meia satuação a 50% de pressão de O_2 (constante de dissociação K_{50} de 2.8 mmHg).

$$y = \frac{pO_2}{K_{50} + pO_2} \tag{4.1}$$

Por outro lado, o valor de K_{50} para a hemoglobina é de 26 mmHg, mas sua função exprime-se de forma diferente à da mioglobina:

$$y = \frac{pO_2^{nH}}{K_{50}^{nH} + pO_2^{nH}} \tag{4.2}$$

Nessa Equação 4.2, *nH* representa o coeficiente de cooperatividade de Hill, que resume a energia distribuida entre as quatro constantes microscópicas de dissociação de O_2 aos quatro centros porfirínicos da hemoglobina (grupos *heme*). Simulando ambas as curvas:

```
K50 <- 2.8
curve(x / (K50 + x),
  xlim = c(0, 100),
  xlab = "pO2 (mmHg)", ylab = "y", lty = "dotted"
)

K50 <- 26
nH <- 2.8
curve(x^nH / (K50^nH + x^nH),
  xlim = c(0, 100),
  xlab = "pO2 (mmHg)", ylab = "y", col = "red",
  add = TRUE
) # "add" permite adicionar curvas ao gráfico
abline(0.5, 0, lty = 2) # acrescenta linha de base em meia saturação
```

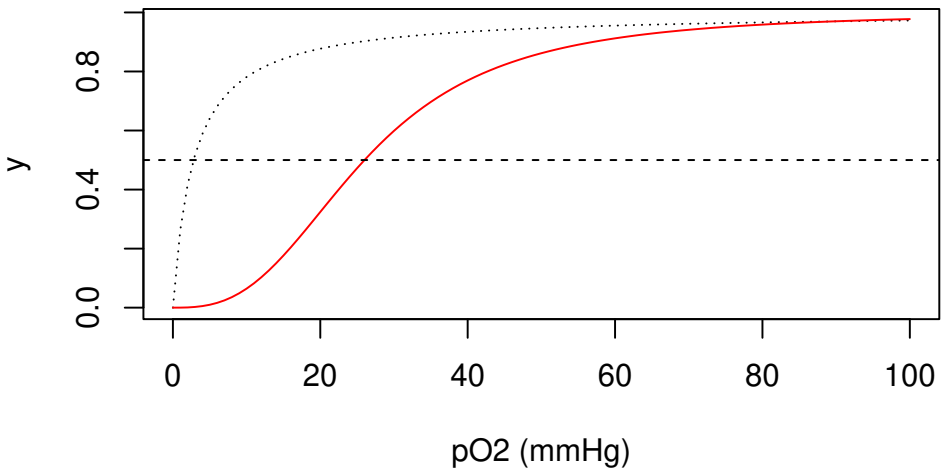


Figura 4.1: Isoterma de saturação de oxigênio à mioglobina (linha contínua) e hemoglobina (linha pontilhada), indicando o intercepto em pO2 de 50% (meia saturação).

Como mencionado no parágrafo anterior, aos quatro centros de ligação com oxigênio molecular reportam-se quatro constantes microscópicas de equilíbrio de dissociação, de *K1* a *K4*. Simplificando o valor de pO_2 para L, ligante, é possível também representar a ligação de O_2 à hemoglobina pela equação de Adair (Pauling 1935):

$$y = \frac{K1 * L + 2 * K2 * K1 * L^2 + 3 * K3 * K2 * K1 * L^3 + 4 * K4 * K3 * K2 * K1 * L^4}{4 * (1 + K1 * L + 2 * K2 * K1 * L^2 + 3 * K3 * K2 * K1 * L^3 + 4 * K4 * K3 * K2 * K1 * L^4)} \tag{4.3}$$

Ocorre que existe um efeito estatístico associado à interação em estudo, já que o O_2 possui 4 sítios iniciais de interação à hemoglobina (Tyuma, Imai, e Shimizu 1973), valores que reduzem até a saturação dos 4 sítios. Dessa forma, é necessário contabilizar as constantes microscópicas *Ki* em razão desse comprometimento estatístico:

$$Ki_{corr} = \frac{i}{N - 1 + i} * Ki \tag{4.4}$$

No R, isso pode ser auxiliado por um *loop for*:

```
K <- c(0.011, 0.016, 0.118, 0.400) # vetor de constantes microscópicas de
# dissociação de Hb para O2
L <- seq(1, 201, 2) # vetor de teores de O2

Kcorr <- c() # inicializa um vetor vazio para saída do vetor corrigido de Ki
N <- 4 # declara o número de sítios na Hb
for (i in 1:N) Kcorr[i] <- i / (N - i + 1) * K[i]
Kcorr # apresenta o vetor de valores de Ki corrigidos para o efeito estatístico
```

```
[1] 0.003 0.011 0.177 1.600
```

Perceba que os valores para *Ki* corrigidos estão em proporção que segue a disponibilidade de sítios, de 4 vezes menor para o 1o. sítio (maior ligação), até 4 vezes maior para o 4o. sítio (menor ligação). Agora é possível aplicar-se a Equação 4.3 utilizando-se as constantes calculadas como segue:

```
numer <- K[1] * L + 2 * K[2] * K[1] * L^2 + 3 * K[3] * K[2] * K[1] * L^3 +
  4 * K[4] * K[3] * K[2] * K[1] * L^4
denom <- 1 + numer
y <- numer / denom
plot(L, y, xlab = "pO2", type = "l", col = 2)
```

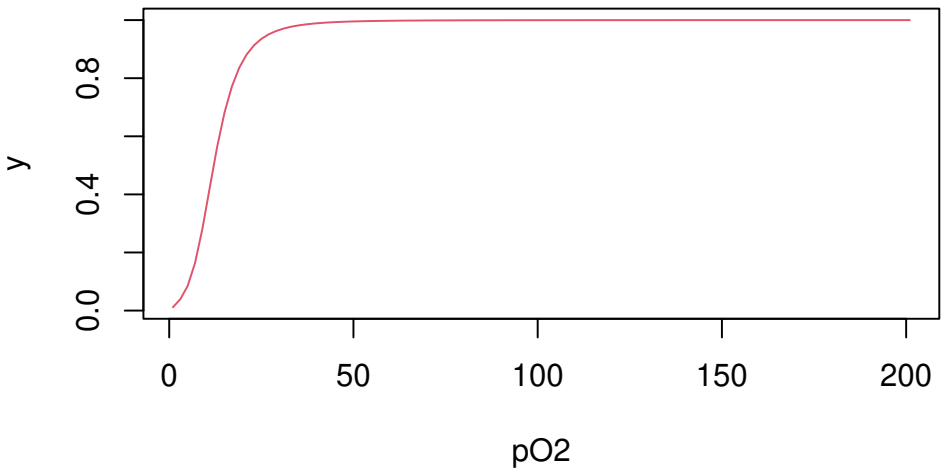


Figura 4.2: Isoterma de saturação de oxigênio à hemoglobina pela equação de Adair.

Por fim, também é possível obter a curva de ligação utilizando-se uma variação de *loop for* na qual a Equação 4.3 é produzida por iteração, como segue (Bloomfield 2009):

```
# Cálculo de y em cada L
Yi <- function(L, Kcorr) {
  N <- length(Kcorr)
  conc <- c()
  conc[1] <- L * Kcorr[1]
  for (i in 2:N) conc[i] <- conc[i - 1] * L * Kcorr[i]
  numer2 <- sum((1:N) * conc) / N
  denom2 <- 1 + sum(conc)
  return(numer2 / denom2)
}

# Cálculo de y para o vetor de L
```

```
Y <- function(L, Kcorr) {
  YY <- c()
  for (j in 1:length(L)) YY[j] <- Yi(L[j], Kcorr)
  return(YY)
}

#  Aplicação da função de y para L e gráfico
Yfinal <- Y(L, Kcorr)
plot(L, Yfinal, type = "l", col = 2, xlab = "pO2", ylab = "y")
```

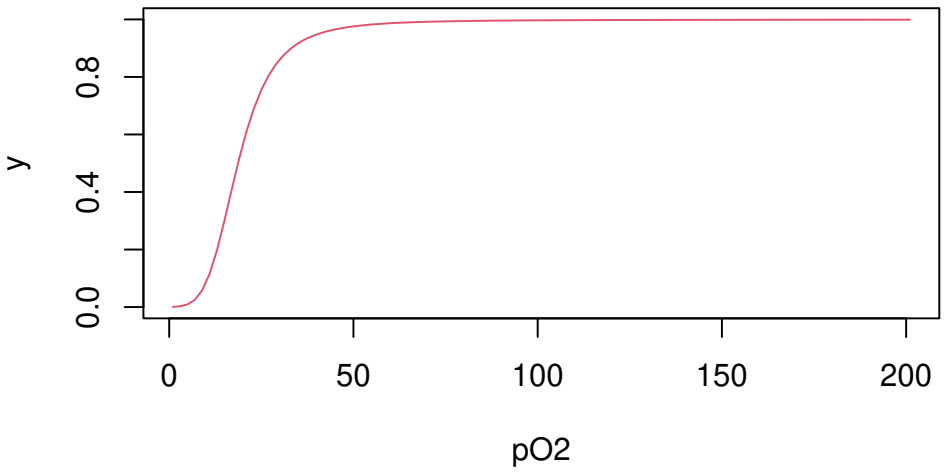


Figura 4.3: Curva de saturação de oxigênio à hemoglobina obtida por iteração da equação de Adair, tal como corrigida para o efeito estatístico.

Observe que há mínimas diferenças entre as curvas obtidas pelos distintos procedimentos, em razão dos diferentes algoritmos utilizados.

Sec 4.4 Alguns pacotes do R para estudo de proteínas

Alguns pacote do R são bastante úteis no estudo de proteínas, em especial a biblioteca **seqinr** vista no capítulo Capítulo 3, e que computa diversos valores e informações para sequências proteicas, tais como *pI*, index de hidroxipatia, distribuição de resíduos, entre outros. O sítio do projeto [2] contém informação detalhada para seu uso. Utilizando-se o mesmo procedimento para obtenção da sequência *FASTA* para a lisozima do capítulo Capítulo 3 (código CAA32175 no sítio *NCBI*), pode-se obter um conjunto extenso de informações da proteína, como exemplificado abaixo:

```
library(seqinr)
lisozima <- c("KVFERCELARTLKRLGMDGYRGISLANWMCLAKWESGYNTRATNYNAGDRSTDYGIFQ
INSRYWCNDGKTPGAVNACHLSCSALLQDNIADAVACAKRVVRDPQGIRAWVAWRNRCQNRDVRQYVQGCGV")
seq_liso <- s2c(lisozima) # converte sequência de string de aminoácidos para
# o padrão do seqinr (vetor de caracteres)
seq_liso2 <- seq_liso[seq_liso != "\n"] # eliminação de espaços exigida pelo
 # seqinr advindos do procedimento de copiar/colar.
seq_liso2
```

```
  [1] "K" "V" "F" "E" "R" "C" "E" "L" "A" "R" "T" "L" "K" "R" "L" "G" "M" "D"
```

[2] Sítio do projeto Seqinr: http://seqinr.r-forge.r-project.org/

```
 [19] "G" "Y" "R" "G" "I" "S" "L" "A" "N" "W" "M" "C" "L" "A" "K" "W" "E" "S"
 [37] "G" "Y" "N" "T" "R" "A" "T" "N" "Y" "N" "A" "G" "D" "R" "S" "T" "D" "Y"
 [55] "G" "I" "F" "Q" "I" "N" "S" "R" "Y" "W" "C" "N" "D" "G" "K" "T" "P" "G"
 [73] "A" "V" "N" "A" "C" "H" "L" "S" "C" "S" "A" "L" "L" "Q" "D" "N" "I" "A"
 [91] "D" "A" "V" "A" "C" "A" "K" "R" "V" "V" "R" "D" "P" "Q" "G" "I" "R" "A"
[109] "W" "V" "A" "W" "R" "N" "R" "C" "Q" "N" "R" "D" "V" "R" "Q" "Y" "V" "Q"
[127] "G" "C" "G" "V"
```

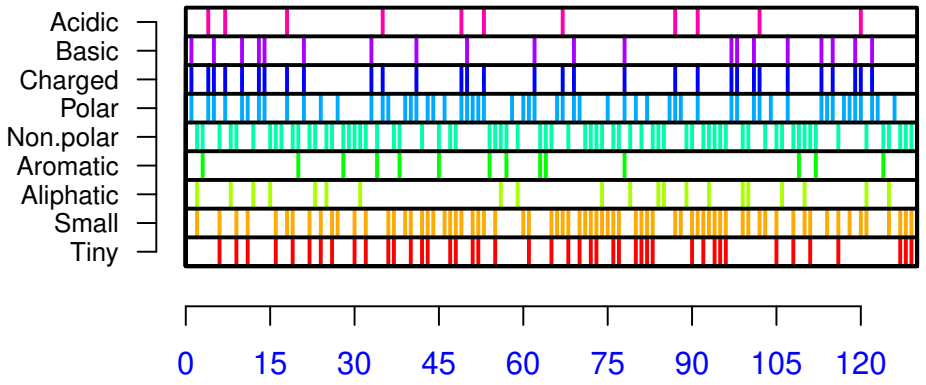
```
pmw(seq_liso2) # peso molecular da proteína
```

```
[1] 14701
```

```
aaa(seq_liso2) # distribuição de resíduos
```

```
  [1] "Lys" "Val" "Phe" "Glu" "Arg" "Cys" "Glu" "Leu" "Ala" "Arg" "Thr" "Leu"
 [13] "Lys" "Arg" "Leu" "Gly" "Met" "Asp" "Gly" "Tyr" "Arg" "Gly" "Ile" "Ser"
 [25] "Leu" "Ala" "Asn" "Trp" "Met" "Cys" "Leu" "Ala" "Lys" "Trp" "Glu" "Ser"
 [37] "Gly" "Tyr" "Asn" "Thr" "Arg" "Ala" "Thr" "Asn" "Tyr" "Asn" "Ala" "Gly"
 [49] "Asp" "Arg" "Ser" "Thr" "Asp" "Tyr" "Gly" "Ile" "Phe" "Gln" "Ile" "Asn"
 [61] "Ser" "Arg" "Tyr" "Trp" "Cys" "Asn" "Asp" "Gly" "Lys" "Thr" "Pro" "Gly"
 [73] "Ala" "Val" "Asn" "Ala" "Cys" "His" "Leu" "Ser" "Cys" "Ser" "Ala" "Leu"
 [85] "Leu" "Gln" "Asp" "Asn" "Ile" "Ala" "Asp" "Ala" "Val" "Ala" "Cys" "Ala"
 [97] "Lys" "Arg" "Val" "Val" "Arg" "Asp" "Pro" "Gln" "Gly" "Ile" "Arg" "Ala"
[109] "Trp" "Val" "Ala" "Trp" "Arg" "Asn" "Arg" "Cys" "Gln" "Asn" "Arg" "Asp"
[121] "Val" "Arg" "Gln" "Tyr" "Val" "Gln" "Gly" "Cys" "Gly" "Val"
```

```
AAstat(seq_liso2, plot = TRUE) # gráfico de distribuição, composição
```




```
$Compo

 *  A  C  D  E  F  G  H  I  K  L  M  N  P  Q  R  S  T  V  W  Y
 0 14  8  8  3  2 11  1  5  5  8  2 10  2  6 14  6  5  9  5  6

$Prop
$Prop$Tiny
[1] 0.3

$Prop$Small
[1] 0.6
```

```
$Prop$Aliphatic
[1] 0.2

$Prop$Aromatic
[1] 0.1

$Prop$Non.polar
[1] 0.6

$Prop$Polar
[1] 0.4

$Prop$Charged
[1] 0.2

$Prop$Basic
[1] 0.2

$Prop$Acidic
[1] 0.08


$Pi
[1] 9
# e proporção de resíduos, valor de pI
```

Entre outras funções constantes do `seqinr` inclui-se a conversão de aminoácidos para abreviações de 1 e 3 letras (`ae` `aaa`, respectivamente), listagem de 544 propriedades físico-químicas dos 20 aminoácidos proteicos (`aaindex`),`pK` (autoexplicativo, e visto anteriormente), e cômputo isolado de *pI* (`computePI`) e de massa molecular (`pmw`), além de várias outras, tanto para proteômica como para genômica.

Outro pacote do R interessante para estudo de proteínas é o *Peptides* [3], que também computa diversas propriedades físico-químicas para sequências de aminoácidos, além de possibilitar a integração de plotagem com o pacote de dinâmica molecular *GROMACS*. Como para o `seqinr`, o `Peptides` necessita de conversão da sequência em *string* para o padrão vetorial reconhecido. Entre as funções do pacote destacam-se o cômputo de 66 descritores para cada aminoácido de uma sequência (`aaDescriptors`), a composição da sequência por classificação dos resíduos (`aaComp`), o cômputo de índice alifático (`aIndex`), o índice de hidrofobicidade (`hydrophobicity`), índice de instabilidade (`instalIndex`), relação de massa/carga (`mz`), massa molecular (`mw`), e pI (`pI`), entre outros.

Entre pacotes mais direcionados ao estudo comparativo e visualização de estruturas, bem como para descritores de bioinformática e quimiogenômica vale mencionar `Bio3d`, `Autoplotprotein`, `protr`, `BioMedR`, e `UniprotR`, entre muitos.

[3] Pacote Peptides: https://cran.r-project.org/web/packages/Peptides/index.html

De modo geral, enzimas são estudadas sob diversos pontos de vista, tais como sua estrutura, mecanismo de ação, e comportamento cinético. Este capítulo visa trabalhar nesse último, com auxílio do R.

De modo geral, a equação simplificada que descreve a atividade de uma enzima E sobre um substrato S pode ser descrita como:

$$E + S \underset{k2}{\overset{k1}{\rightleftharpoons}} ES \overset{k3}{\rightarrow} E + P \tag{5.1}$$

Onde P representa o produto da reação, ES o complexo ativado no estado de transição, e *k1*, *k2* e *k3* as constantes de velocidade da reação.

Pela aproximação de Briggs-Haldane para o estado estacionário, e o tratamento de Henri-Michaelis-Menten, a equação que define a curva de velocidade da reação enzimática em função do teor de substrato pode ser descrita abaixo:

$$v = \frac{Vm * S}{Km + S} \tag{5.2}$$

Onde *Km* representa a *constante de Michaelis-Menten*, e *Vm* a velocidade limite da reação (por vezes denominada erroneamente como *velocidade máxima*, embora a hipérbole quadrática descrita pela função não exiba valor máximo por não atingir uma assíntota). Por sua vez *Km* pode ser definido a partir das constantes de velocidade da Equação 5.1 como:

$$Km = \frac{k1 + k3}{k2} \tag{5.3}$$

Portanto, dados os valores de *Vm* e *Km*, podemos descrever um *comportamento de Michaelis-Menten* para uma enzima como:

```
#  Curva cinética de Michaelis-Mentem

Vm <- 10
Km <- 0.5
curve(Vm * x / (Km + x),
  xlim = c(0, 10),
  xlab = "[S]", ylab = "v"
)
abline(h = 5, lty = 2, col = "blue")
abline(v = 0.5, lty = 2, col = "blue")
text(x = 1, y = 0.2, "Km", col = "blue")
text(1, 5.3, "Vm/2", col = "blue")
```

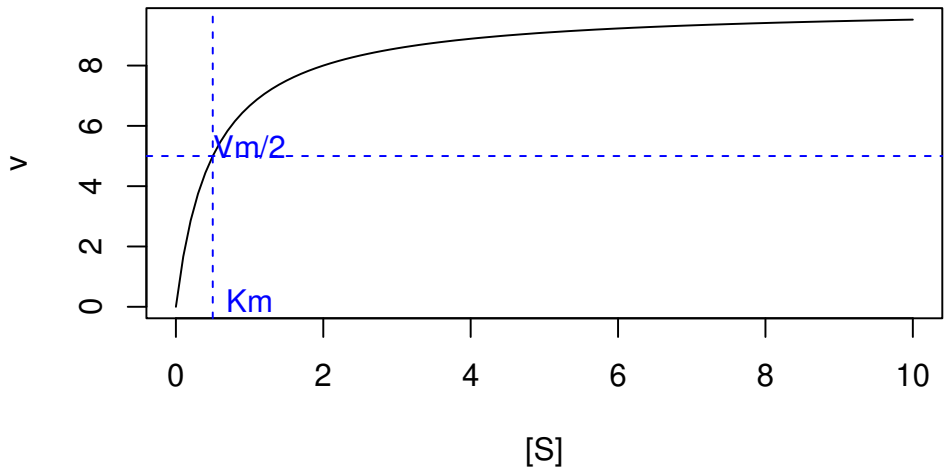


Figura 5.1: Curva de Michaelis-Mentem para uma enzima exibindo Vm=10 e Km=0,5 (50 curvas).

Por essa relação, o valor de *Km* é representado pelo teor de substrato que confere metade do valor de *Vm* para a reação. Observe que pela Equação 5.2 o valor de *Km* no denominador influencia inversamente a velocidade *v* da reação; ou seja, quanto maior o valor de *Km*, menor a taxa da reação enzimática. Isto pode ser ilustrado iterativamente com um *loop*, tal como segue:

```
Vm <- 10
Km <- seq(from = 0.1, to = 10, by = 0.2) # sequência para 50 valores de Km
for (i in 1:length(Km)) { # loop para adicionar curva de Michaelis-Mentem
  #a cada valor de Km
  add <- if (i == 1) FALSE else TRUE # controle de fluxo que permite adição
  # de curva a partir da segunda iteração (ou seja, quando i > 1)
  curve(Vm * x / (Km[i] + x),
    col = i, lwd = 0.8, from = 0, to = 10, n = 100,
    xlab = "[S}", ylab = "v", add = add
  )
}
arrows(0.5, 9, 3, 6, length = 0.1, angle = 45, col = "blue") # seta para Km
text(0.2, 9, "Km", col = "blue") # indexador para Km
```

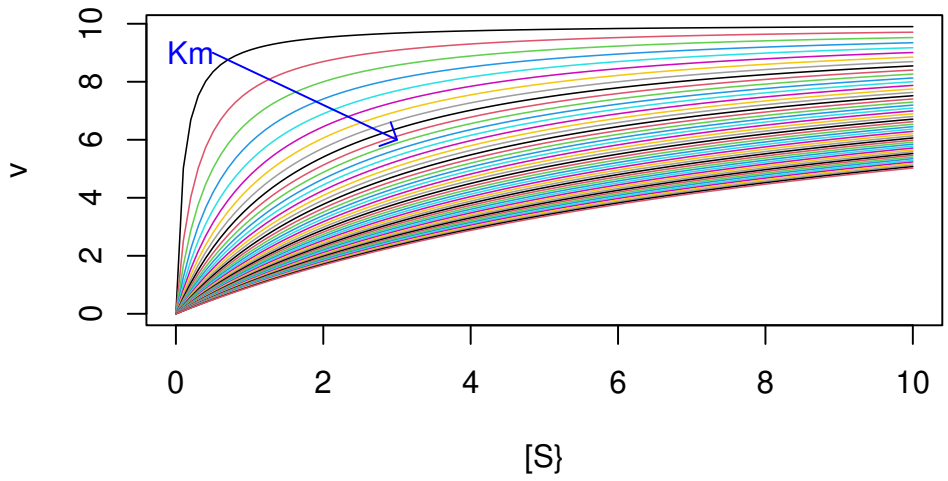


Figura 5.2: Curvas de Michaelis-Menten com variação iterativa para Km de 0.1 a 10.

As curvas representadas nas figuras Figura 5.1 e Figura 5.2) foram produzidas com a Equação 5.2, sendo possível extrair os parâmetros cinéticos *Vm* e *Km* facilmente, já que não há desvios computados para a *velocidade inicial* da reação. Se, por outro lado, estivermos diante de valores experimentais de uma catálise de comportamento de michaelinano e desejarmos extrair os parâmetros cinéticos, melhor será ajustar a equação não-linear de Michaelis-Menten diretamente (algoritmos como Gauss-Newton, Simplex, Levenberg-Marquadt) ou, de modo mais simples, transformar as variáveis S e v de tal modo que permitam um ajuste linear por mínimos quadrados.

Não obstante, a função que descreve a equação de Michaelis-Mentem constitui uma hipérbole quadrática e, como tal, não possui assíntota matemática, apenas visual. De fato, os próprios autores do trabalho original, Leonor Michaelis e Maud Mentem, reportaram seus dados com a representação de S em eixo logaritmo permitindo melhor visualização da região assintótica do gráfico (Michaelis e Menten 1913).

Sec 5.1 Obtenção de parâmetros cinéticos a partir de dados expermentais simulados

Para a determinação dos parâmetros cinéticos *Vm* e *Km* obtidos a partir de uma simulação experimental de dados de S e v, é necessário em primeiro lugar obter-se os pontos experimentais, tal como segue:

```
Vm <- 10
Km <- 0.5
set.seed(1500) # fixa a semente para geração de dados aleatórios reproduzíveis
erro <- runif(20, 0, 1) # comando para erro uniforme (no. de pontos, min, max)
curve(Vm * x / (Km + x) + erro,
  type = "p", from = 0, to = 1, n = 20,
  xlab = "[S}", ylab = "v"
) # elaboração da curva com cômputo de erro uniforme
```

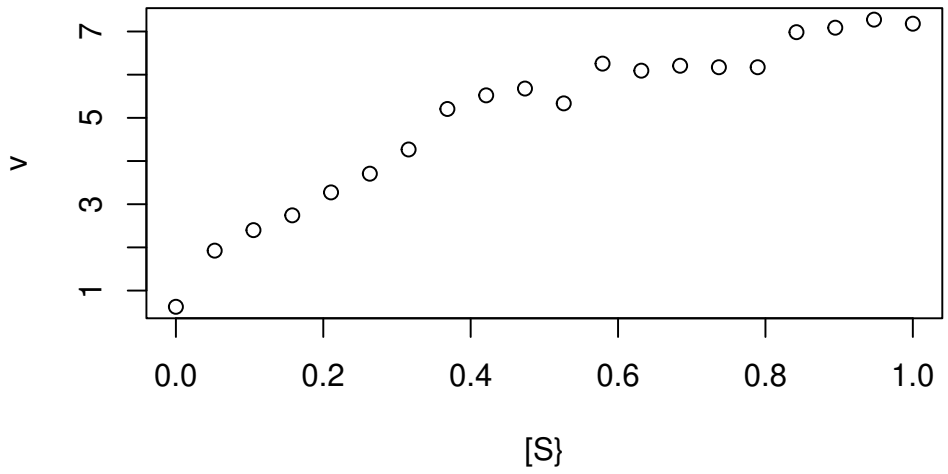


Figura 5.3: Simulação de pontos experimentais (n=20) obtidos a partir da equação de Michaelis-Menten.

Perceba que pela Figura 5.3, já não é mais possível definir-se uma região assintótica que permita a determinação de *Vm* e, por consequência, de *Km*. Nesse caso, pode-se obter os parâmetros cinéticos por tranformação da função hiperbólica de Michaelis-Mentem para uma função linear correlata, sucedendo-se o ajuste linear dos dados transformados para a obtenção dos parâmetros de catálise.

Sec 5.2 Linearizações e ajustes

Diversas são as linearizações encontradas na literatura para a equação de Michaelis-Mentem. Para exemplificá-las, segue um trecho de código contendo as quatro mais empregadas. Para isso utilizou-se um par de funções do `R` para, respectivamente, estabelecer a área gráfica e sua subdivisão para plotagem em 4 paineis (`par` e `mfrow` ou `mfcol`):

```
S <- c(0.1, 0.2, 0.5, 1, 5, 10, 20) # cria um vetor para substrato
Km <- 0.5
Vm <- 10 # estabelece os parâmetros enzimáticos
v <- Vm * S / (Km + S) # aplica a equação de MM ao vetor de S
par(mfrow = c(2, 2)) # estabelece área de plot pra 4 gráficos
plot(S, v, type = "o", main = "Michaelis-Mentem")
plot(1 / S, 1 / v, type = "o", main = "Lineweaver-Burk")
plot(v, v / S, type = "o", main = "Eadie-Hofstee")
plot(S, S / v, type = "o", main = "Hanes-Woolf")
layout(1) # volta à janela gráfica normal
```

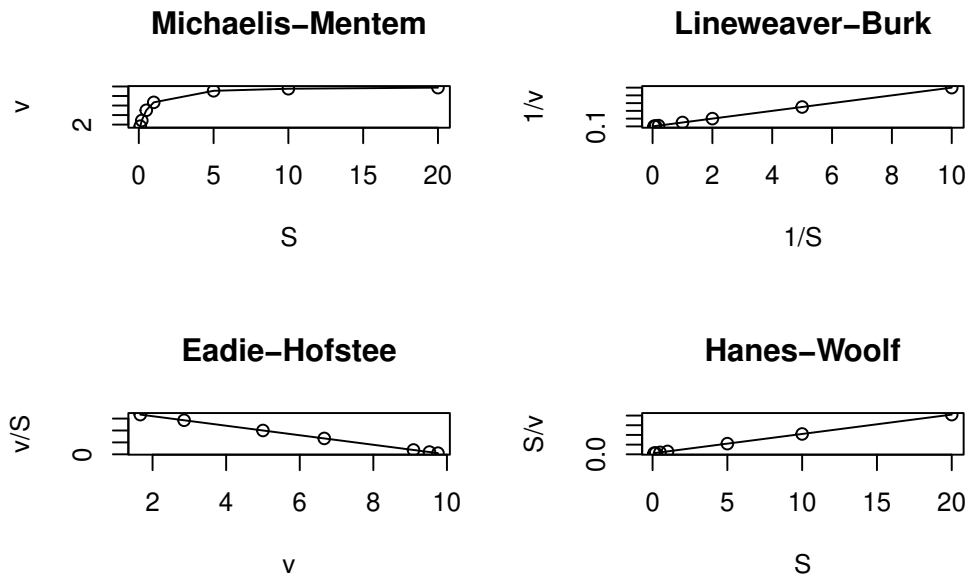


Figura 5.4: Principais linearizações da equação de Michaelis-Menten.

Dentre essas transformações lineares para cinética enzimática, as mais frequentemente encontradas são de *Lineweaver-Burk* (ou duplo-recíproco), e a de *Eadie-Hofstee*, sendo a primeira a mais comum na literatura. No entanto, a aplicação do formalismo de Eadie-Hofstee para interação ligante-proteína também é a mais reportada nessa área, embora seja tratada como representação de Scatchard (Scatchard 1949).

5.2.1 Linearização por transformação de Lineweaver-Burk

A forma linear para a equação de Linewaver-Burk é obtida a partir da transformação da equação de Michaelis-Menten que segue:

$$\frac{1}{v} = \frac{1}{Vm} + \frac{Km}{Vm} * \frac{1}{S} \tag{5.4}$$

Dessa forma, os dados obtidos pela Figura 5.3 são transformados para seu duplo-recíproco, resultando em:

```
S <- seq(0.1, 1, length.out = 20) # gera uma sequência com 20 pontos entre
# 0 e 1 para valores de substrato
Vm <- 10
Km <- 0.5 # parâmetros cinéticos
set.seed(1500) # estabelecer a mesma semente aleatória do gráfico direto
 # de Michaelis-Menten, para reproducibilidade dos pontos
erro <- runif(20, 0, 1) # comando para erro uniforme (no. de pontos, min, max)
v <- Vm * S / (Km + S) + erro # equação de Michaelis-Menten

inv.S <- 1 / S # cria variáveis para o duplo-recíproco
inv.v <- 1 / v
plot(inv.v ~ inv.S, xlab = "1/S", ylab = "1/v") # elabora o gráfico de
 # Lineweaver-Burk
```

5

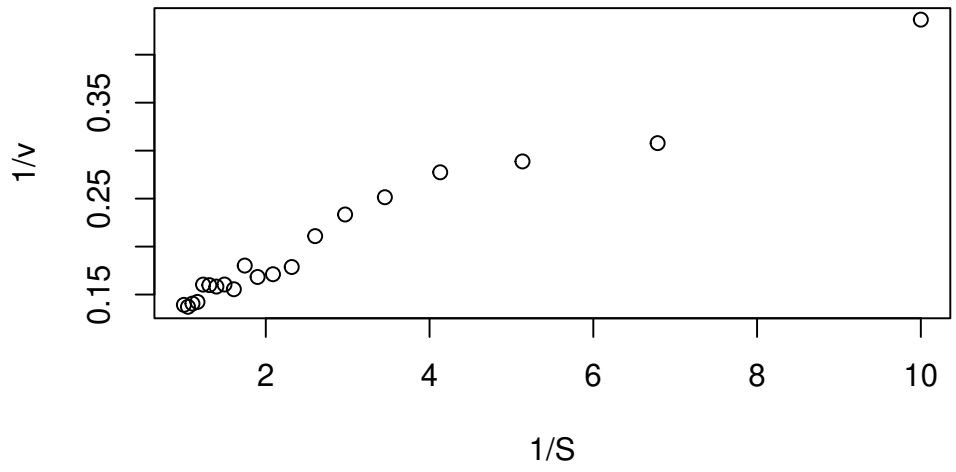


Figura 5.5: Representação de Lineweaver-Burk para os dados simulados da curva de Michaelis-Menten.

Observe que a Figura 5.5 apresenta agora um uma distribuição de valores que possibilita seu ajuste linear por mínimos quadrados (regressão linear). No R, isso pode ser facilmente conduzido pelo trecho de código (*chunk*) que segue:

```
reg.LB <- lm(inv.v ~ inv.S) # expressão para ajuste linear
summary(reg.LB) # resultados do ajuste
```

```
Call:
lm(formula = inv.v ~ inv.S)

Residuals:
      Min        1Q    Median        3Q       Max
-0.028198 -0.009858 -0.003496  0.007482  0.028416

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.113634   0.005147   22.08 1.74e-14 ***
inv.S       0.032772   0.001461   22.42 1.33e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.01459 on 18 degrees of freedom
Multiple R-squared:  0.9654,    Adjusted R-squared:  0.9635
F-statistic: 502.8 on 1 and 18 DF,  p-value: 1.325e-14
```

```
plot(inv.v ~ inv.S, xlab = "1/S", ylab = "1/v") # gráfico de Lineweaver-Burk
abline(reg.LB, col = "blue") # sobreposição do ajuste ao gráfico
```

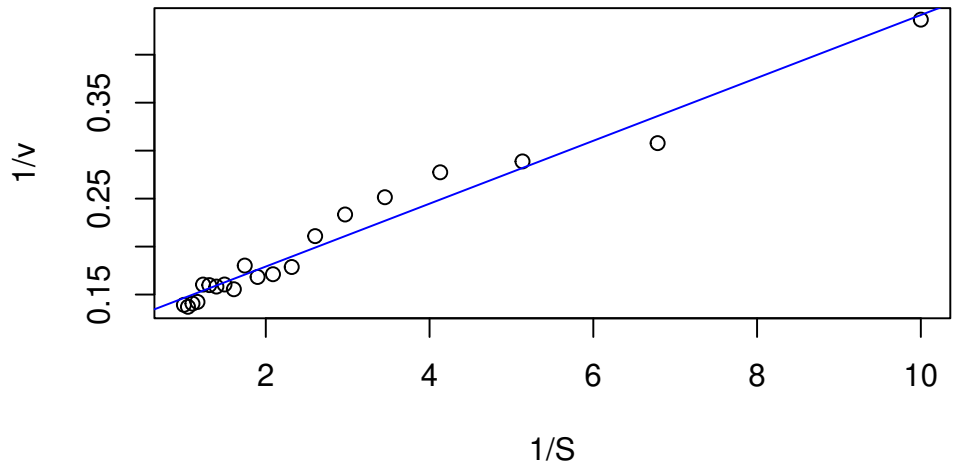


Figura 5.6: Ajuste linear para os dados de Lineweaver-Burk.

A tabela produzida pelo `R` para a função `lm` de ajuste linear por mínimos quadrados possui diversas informações que nos permite avaliar a qualidade da regressão. Brevemente, esse tabela nos fornece o valor de cada parâmetro do ajuste conforme a equação que segue:

$$y = a + b * x \tag{5.5}$$

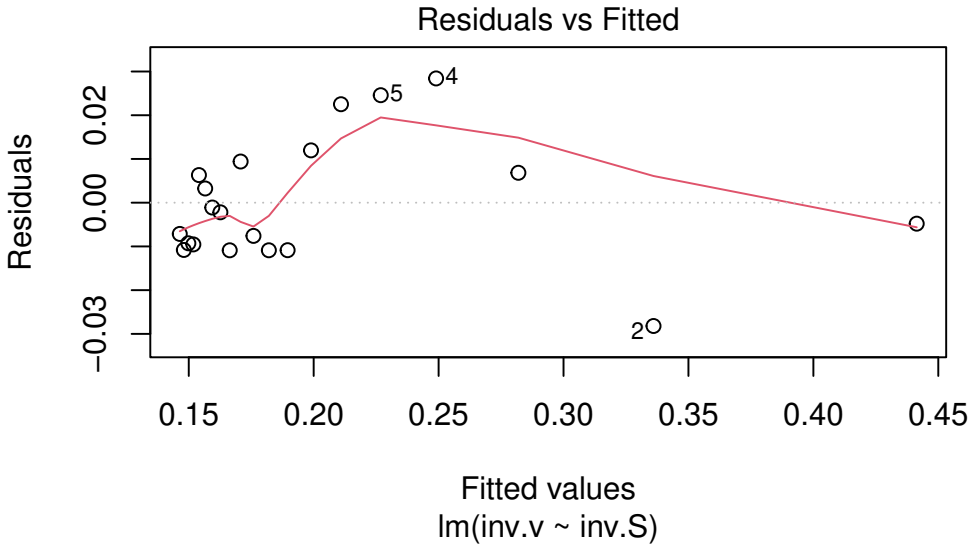
Nesse caso, a refere-se ao intercepto presente na tabela (*(intercept)*, ou 1/Vm) e b à inclinação (*inv.S*, ou Km/Vm). Além disso, a tabela também fornece diversos resultados complentares, elencados a seguir:

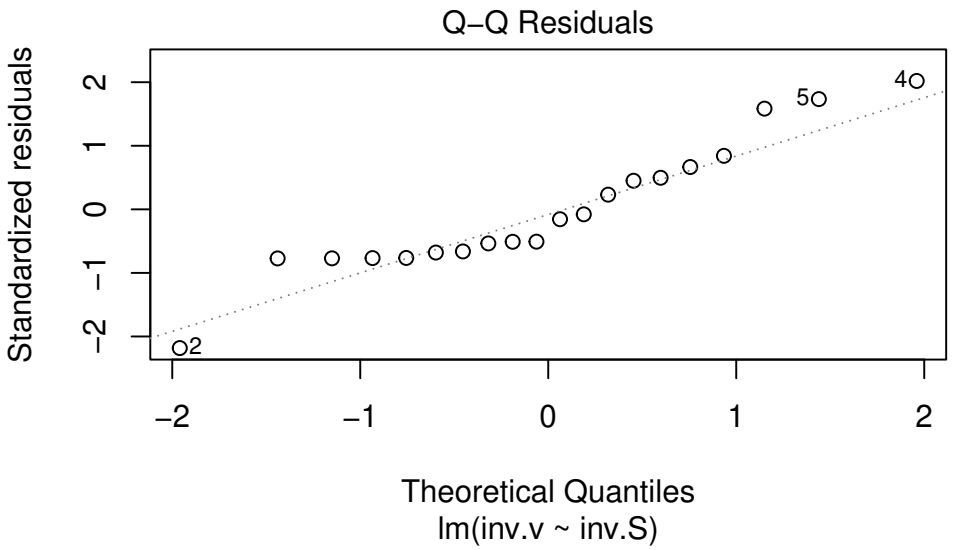
1. valor de erro-padrão dos parâmetros (*Std. Error*);
2. valor da distribuição t de Student (t *value*);
3. o respectivo nível de probabilidade (*Pr*) com indicação de significância (asteriscos);
4. erro padrão residual (*Residual standard error*);
5. valor dos coeficientes de determinação bruto (*Multiple R-squared*) e ajustado para os graus de liberdade (*Adjusted R-squared*);
6. valor da distribuição F de Snedocor (*F-statistic*) de variância do ajuste;
7. graus de liberdade (*DF*) e o valor de significância da regressão ao modelo linear obtido pela análise de variância (*p-value*).

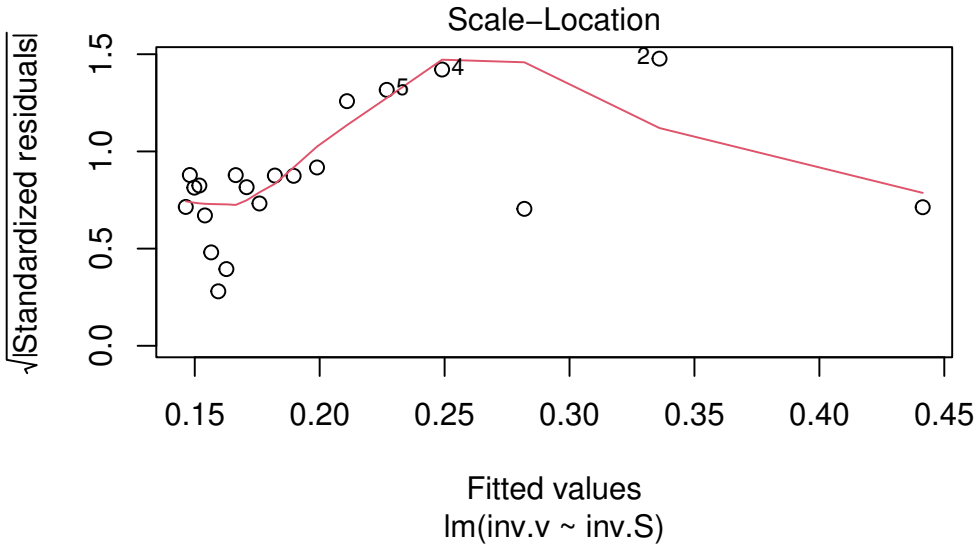
Não obstante, a qualidade do ajuste linear também pode ser verificada pela produção de gráficos diagnósticos estatísticos, bastando-se aplicar o trecho simples abaixo:

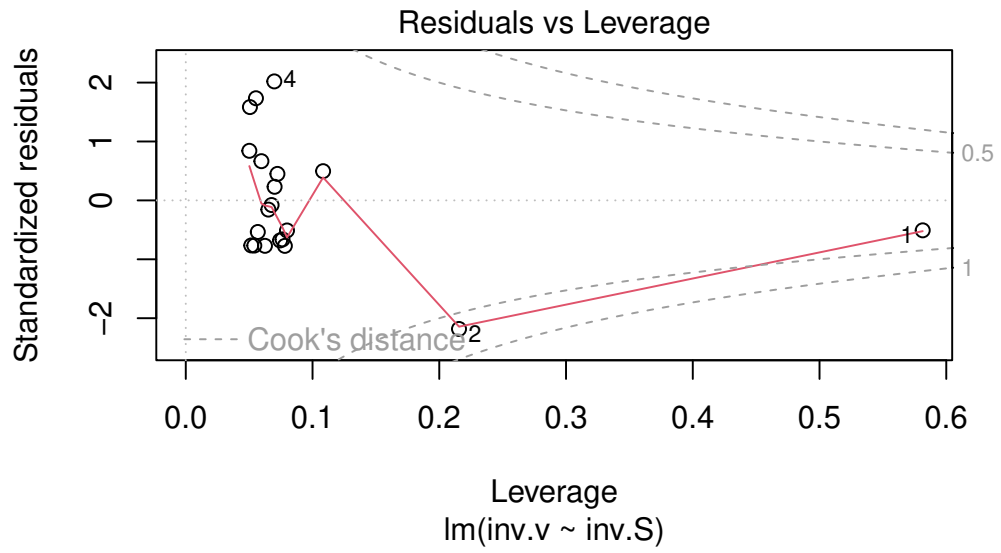
```
plot(reg.LB) # comando para geração de gráficos diagnósticos de ajuste linear
```

5


Residuals vs Fitted
Residuals
0.02
0.00
−0.03
4
5
2
0.15
0.20
0.25
0.30
0.35
0.40
0.45
Fitted values
lm(inv.v ~ inv.S)


Q−Q Residuals
Standardized residuals
2
1
0
−1
−2
4
5
2
−2
−1
0
1
2
Theoretical Quantiles
lm(inv.v ~ inv.S)





Esses gráficos diagnósticos também podem ser alocados em paineis, como ilustrado para as linearizações de Michaelis-Mentem acima.

```
reg.LB <- lm(inv.v ~ inv.S)
par(mfrow = c(2, 2))
plot(reg.LB)
layout(1)
```

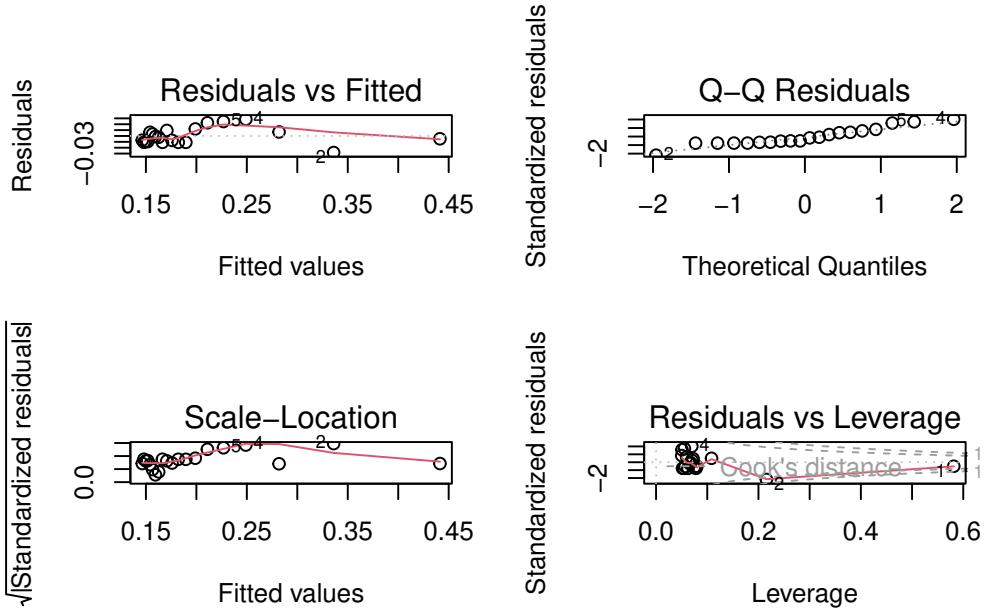
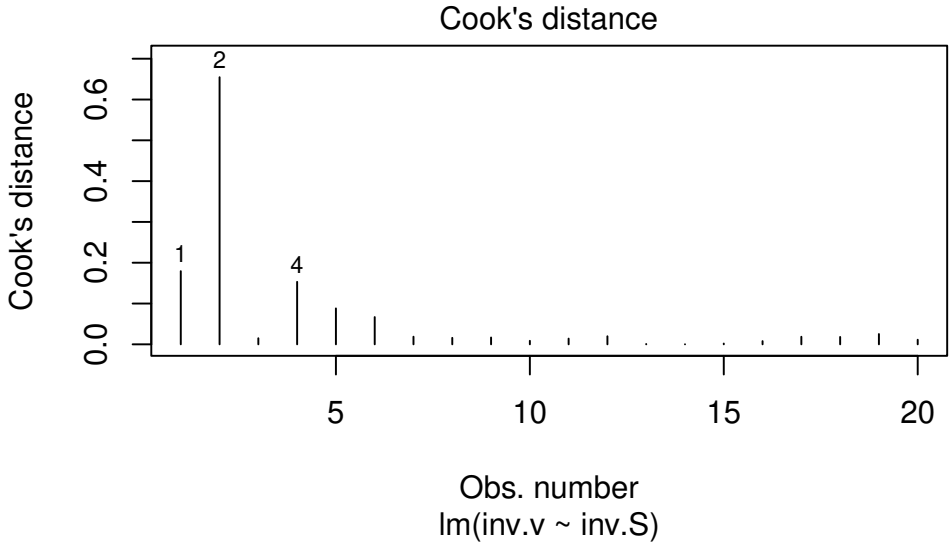


Figura 5.7: Gráficos diagnósticos de ajuste linear.

Os quatro gráficos gerados acima referem-se, respectivamente, 1) à variação de resíduos com os valores ajustados, 2) a um teste de distribuição normal dos resíduos, 3) à variação de resíduos padronizados em função dos valores ajustados, e 4) à observação de valores influenciáveis identificáveis pela *distância de Cook* para cada observação. Em relação à esse último, pode-se opcionalmente definir sua aparência para identificação daqueles valores por seleção (`which`, 4 ou 6, por ex), tal como em:

```
plot(reg.LB, which = 4)
```



Por esses dois procedimentos, tabela e gráficos diagnósticos, é possível aferir a qualidade de um ajuste linear pelo `R`. Em paralelo, diversas são as funções associadas à própria função `lm` para modelos lineares (objetos), o que reforça o caráter de linguagem orientada a objeto do `R`. Entre essas vale citar, com significado intuitivo, `coef`, `fitted`, `predict`, `residuals`, `confint`, e `deviance`.

Para acessar os parâmetros contidos na função `lm`, assim como outras de mesma natureza no `R`, basta digitar `args`:

```
args(lm)
```

```
function (formula, data, subset, weights, na.action, method = "qr",
    model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE,
    contrasts = NULL, offset, ...)
NULL
```

Complementarmente, vale mencionar a existência de inúmeros pacotes do `R` para diversas situações e tratamentos estatísticos de dados para modelos lineares, e que fogem ao escopo deste manuscrito, tais como os que possibilitam análises de *outliers* (valores extremos), *Generalized Linear Models*, *Mixed Effects Models*, *Non-parametric Regression*, entre outros. Entre os pacotes do `R` complementares para regressão linear vale mencionar `car`, `MASS`, `caret`, `glmnet`, `sgd`, `BLR`, e `Lars`.

5.2.1.1 Considerações sobre a linearização por Lineweaver-Burk.

De volta ao estudo da cinética de estado estacionário da catálise enzimática, mencionamos acima a possibilidade de linearizações da equação de Michaelis-Mentem por dois tratamentos mais comuns, Lineweaver-Burk e Eadie-Hofstee. De modo geral, qualquer tratamento que resulte na transformação de um modelo original (equação de Michaelis-Menten, no caso) por linearização resultará em desvios estatísticos.

Analisando a equação de Lineweaver-Burk, por exemplo, pode-se evidenciar que, como os valores de S e v estão representados por seus recíprocos, uma pequena variação em v resultará numa grande variação em $1/v$. Por outro lado, a escolha da faixa de teores de S também é extremamente importante para a extração de parâmetros cinéticos. Isso pode ser ilustrado no exemplo que segue:

```
Vm <- 10
Km <- 0.5
set.seed(1500) # semente fixa para erro aleatório
erro <- runif(length(S), 0, 0.1)
S <- seq(1, 10, 0.1)
v <- Vm * S / (Km + S) + erro
plot(v ~ S, xlab = "S", ylab = "v")
```

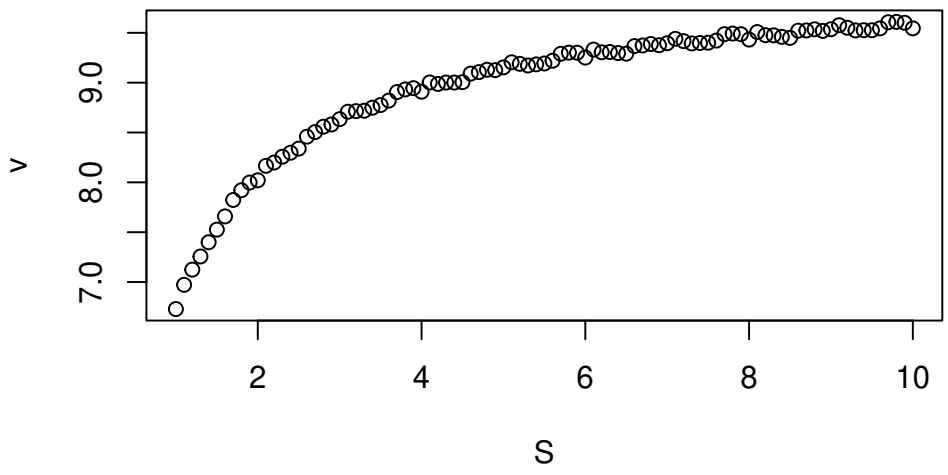


Figura 5.8: Curva de Michaelis-Mentem simulada com erro experimental aleatório. Vm=10; Km=0.5.

Observe que a Figura 5.8 inicia em velocidade próxima a zero, e termina em velocidade próxima à Vm, com teor de $S \gg Km$ (50 vezes, de fato). Essa condição permite uma extração segura dos parâmetros cinéticos, os quais podem ser obtidos a partir do ajuste linear do gráfico duplo-recíproco.

5

```r
# Chunk para Lineweaver-Burk
set.seed(1500) # semente fixa para erro aleatório
erro <- runif(length(S), 0, 0.2)
Vm <- 10
Km <- 0.5 # parâmetros cinéticos
inv.S <- 1 / seq(1, 10, 0.1) # 1/S
inv.v <- 1 / (Vm * S / (Km + S) + erro) # 1/v
plot(inv.S, inv.v)
lm.LB2 <- lm(inv.v ~ inv.S) # ajuste linear
summary(lm.LB2) # resultados do ajuste
```

```
Call:
lm(formula = inv.v ~ inv.S)

Residuals:
       Min         1Q     Median         3Q        Max
-0.0015050 -0.0005613 -0.0001463  0.0007522  0.0014122

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.0991811  0.0001356   731.3   <2e-16 ***
inv.S       0.0481555  0.0004193   114.9   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0007741 on 89 degrees of freedom
Multiple R-squared:  0.9933,    Adjusted R-squared:  0.9932
F-statistic: 1.319e+04 on 1 and 89 DF,  p-value: < 2.2e-16
```

```r
abline(lm.LB2, col = "blue")
```

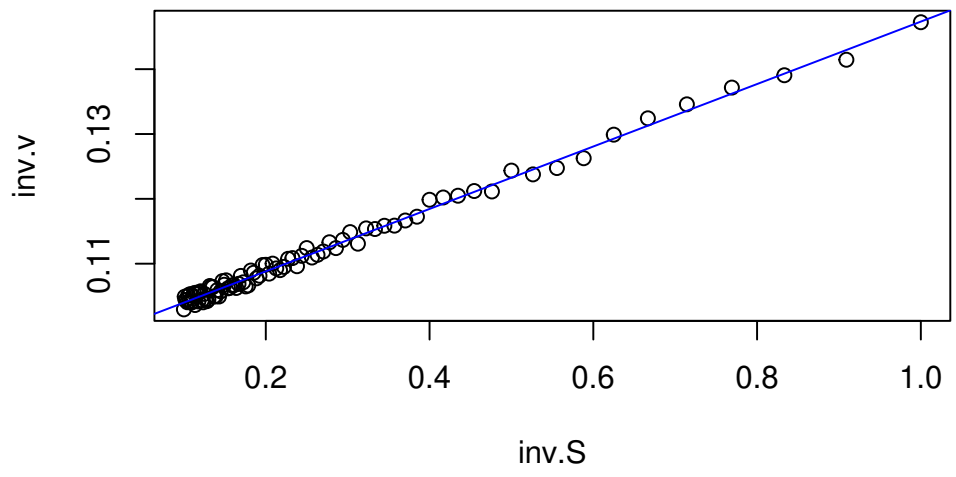


Figura 5.9: Lineweaver-Burk dos dados do gráfico anterior.

Perceba como os valores de *Vm* e *Km* ajustados aproximaram-se dos valores preditos, já que para o gráfico duplo-recíproco:

$$Vm = \frac{1}{intercepto}; Km = intercepto * Vm \tag{5.6}$$

Agora, o que aconteceria se os pontos experimentais estivessem em faixas distintas de teores de S ? Exemplificando, se os pontos fossem coletados em três regiões distintas de S: 10-100, 30-100, e 50-100 ?

Isso pode ser ilustrado variando-se a faixa de valores de S iterativamente, e inspecionando-se o gráfico duplo-recíproco resultante como no trecho de código que segue, e para os mesmos valores da Figura 5.9.

```
# Chunk para duplos-recíprocos de dados simulados com variação em [S]

set.seed(1500) # mesma semente aleatória para reproducibilidade de erro
Vm <- 10
Km <- 0.5 # estabelece os parâmetros de MM
S <- seq(10, 100, 10) # cria-se uma sequência inicial para S
v <- Vm * S / (Km + S) # aplicação equação de MM à S
plot(1 / S, 1 / v, type = "n", ylim = c(0.098, 0.106)) # elabora o
 # duplo-recíproco sem pontos
for (i in 1:3) { # inicia a iteração para gráficos de Lineweaver-Burk
  S <- seq(10 * i, 100, length.out = 100) # gera uma sequência S com
   # 100 pontos, produzindo 5 vetores que iniciam em valores diferentes
   # para S (10, 30 e 50)
  erro <- runif(length(S), 0, 0.1) #  erro para adição à vetor de
   # velocidade inicial, com no. de pontos em função do vetor de S
  add <- if (i == 1) FALSE else TRUE # controle de fluxo para plotagem
   # de pontos no gráfico vazio
  inv.S <- 1 / S
  inv.v <- 1 / ((Vm * S / (Km + S)) + erro) # novos valores para o
  # duplo-recíproco em função da iteração
  points(inv.v ~ inv.S, xlab = "1/S", ylab = "1/v", col = i, add = add)
  # adição de pontos ao gráfico de Lineweaver-Burk, com identificação
   # por cores (1, 2, 3, 4 e 5)
  lm.LB <- lm(inv.v ~ inv.S) # elabora o ajuste linear
  abline(lm.LB, col = i, lty = i) # sobrepõe as linhas de ajuste
}
```

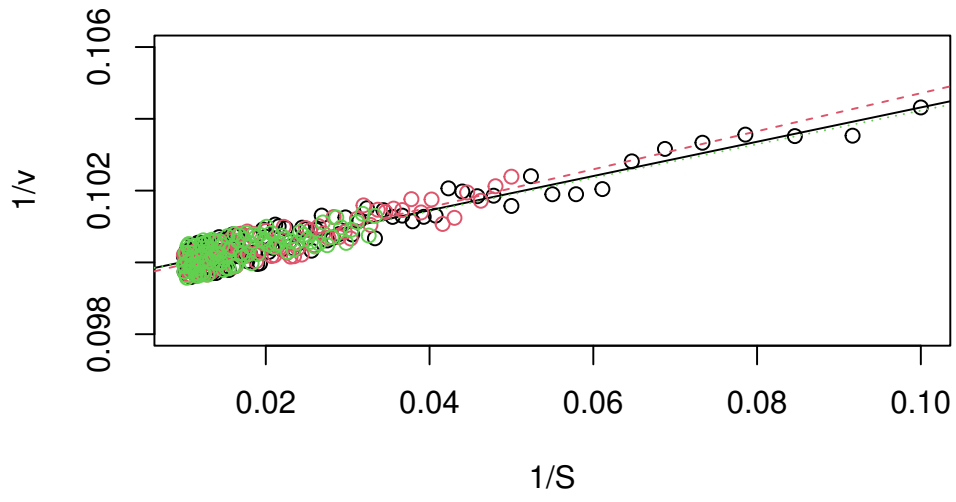


Figura 5.10: Gráficos duplo-recíproco para a curva simulada de Michaelis-Menten, para distintos teores de S inicial.

Observe agora que pela Figura 5.10 resultante, tanto o intercepto como a inclinação obtidos a partir dos ajustes foram dependentes da faixa de seleção de S, o que resulta em distintos valores para Vm e Km. Isto mostra como a seleção da faixa de S para o cálculo de Km e Vm é crucial.

5.2.2 Linearização por transformação de Eadie-Hofstee

Como já mencionado, as duas linearizações da equação de Michaelis-Menten mais comuns referem-se à do subtítulo acima. A equação de linearização de Eadie-Hosftee é dada abaixo:

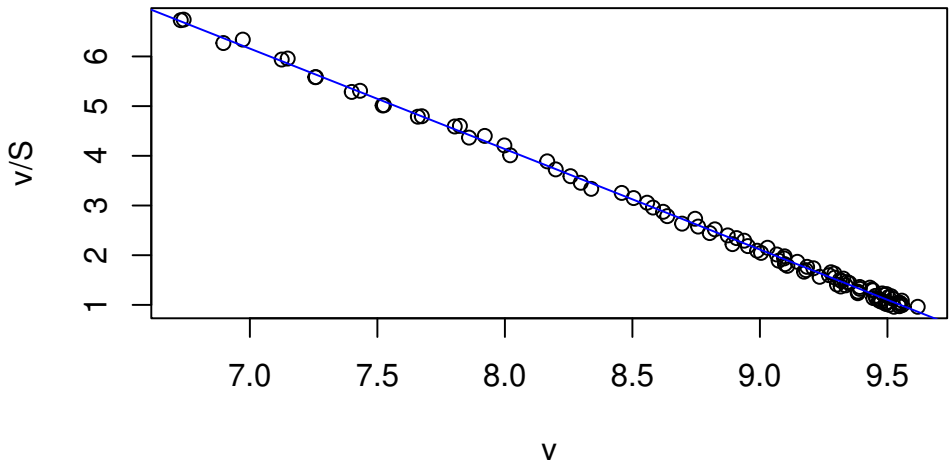
$$v = \frac{1}{Km} * \frac{v}{S} + Vm \tag{5.7}$$

A partir dos dados da Figura 5.3 obtém-se os parâmetros cinéticos diretamente do intercepto (Vm) e da inclinação linear (1/Km) por:

```
# Linearização por Eadie-Hofstee
Vm <- 10
Km <- 0.5
set.seed(1500) # semente fixa para erro aleatório
erro <- runif(length(S), 0, 0.1)
S <- seq(1, 10, 0.1)
v <- Vm * S / (Km + S) + erro
v.S <- v / S
plot(v.S ~ v, xlab = "v", ylab = "v/S")

lm_EH <- lm(v.S ~ v)
summary(lm_EH)
```

```
Call:
lm(formula = v.S ~ v)

Residuals:
     Min       1Q   Median       3Q      Max
-0.11602 -0.06496  0.01172  0.05466  0.12483

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 20.31073    0.07906   256.9   <2e-16 ***
v           -2.02191    0.00893  -226.4   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.07036 on 98 degrees of freedom
Multiple R-squared:  0.9981,    Adjusted R-squared:  0.9981
F-statistic: 5.126e+04 on 1 and 98 DF,  p-value: < 2.2e-16
```

```
abline(lm_EH, col = "blue")
```



5.2.3 Distribuição de erros nas linearizações de Michaelis-Mentem

Embora ambas as representações, Lineweaver-Burk e Eadie-Hofstee, sejam as mais utilizadas e apresentem semelhança na obtenção dos parâmetros cinéticos, sua distribuição de resíduos é bem distinta, assim como as demais transformações lineares de Michaelis-Mentem (Figura 5.4).

O código abaixo ilustra a distribuição de erros dessas transformações, introduzindo uma função importante do `R` para construção de gráficos com *barras de erros*: `arrows`.

```
# Erros aleatórios na eq. de MM e linearizações

Vm <- 10
Km <- 0.5 # fixa os parâmetros de MM
set.seed(1500) # fixa semente para erro aleatório
erro <- runif(length(S), 0, 0.5) # vetor de erro uniforme
S <- c(0.1, 0.2, 0.5, 1, 5, 10, 20) # vetor de substrato
v <- Vm * S / (Km + S) # equação de MM
par(mfrow = c(2, 2)) # área de plot pra 4 gráficos
plot(S, v, type = "o", main = "Michaelis-Mentem")
arrows(S, v, S, v - erro, length = .05, angle = 90) # barra inferior de erro
arrows(S, v, S, v + erro, length = .05, angle = 90) # barra superior de erro

plot(1 / S, 1 / v, type = "o", main = "Lineweaver-Burk")
arrows(1 / S, 1 / v, 1 / S, 1 / (v - erro), length = .05, angle = 90)
arrows(1 / S, 1 / v, 1 / S, 1 / (v + erro), length = .05, angle = 90)

plot(v, v / S, type = "o", main = "Eadie-Hofstee")
arrows(v, v / S, v, (v - erro) / S, length = .05, angle = 90)
arrows(v, v / S, v, (v + erro) / S, length = .05, angle = 90)

plot(S, S / v, type = "o", main = "Hanes-Woolf")
arrows(S, S / v, S, S / (v - erro), length = .05, angle = 90)
arrows(S, S / v, S, S / (v + erro), length = .05, angle = 90)

par(mfrow = c(1, 1)) # retorno à janela gráfica normal
```

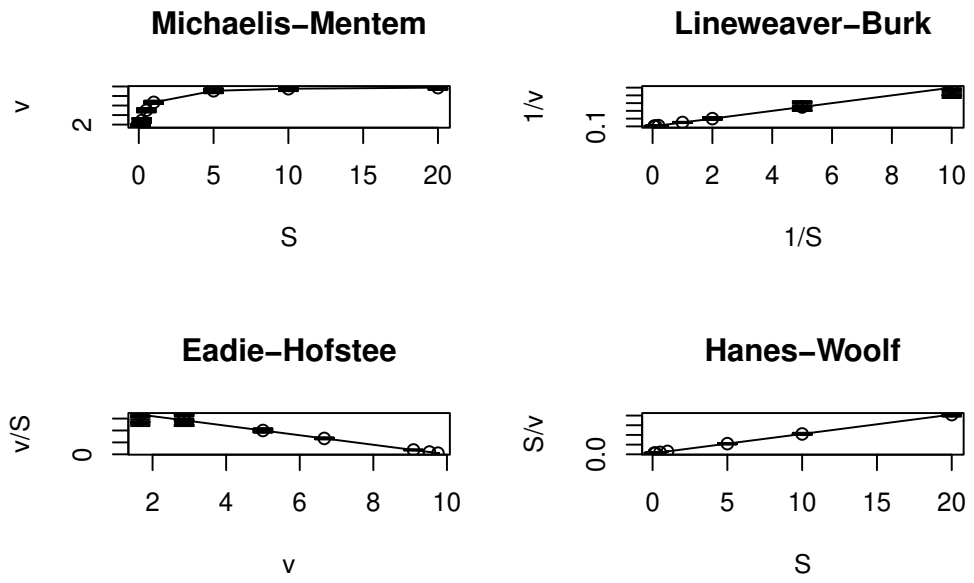


Figura 5.11: Distribuição de erros aleatórios para a equação de Michaelis-Mentem e suas transformações lineares.

Pelo gráfico da Figura 5.11 fica evidente que as transformações de Lineweaver-Burk e de Eadie-Hofstee, ainda que tenham prevalência na literatura, são as que apresentam a maior variabilidade de erro a partir dos dados experimentais, o que converge para uma menor precisão na determinação de parâmetros cinéticos. Além disso, observa-se que a transformação de Hanes-Woolf é a que apresenta a menor dispersão de resíduos entre as demais, igualando-se aos erros experimentais da equação hiperbólica de Michaelis-Mentem. Apesar disso, a linearização de Hanes-Woolf é muito pouco reportada na literatura.

Ainda que a representação dos duplos-recíprocos tenha em si algumas desvantagens, tais como a dispersão de erros principalmente em valores baixos de S, é a única dentre as mencionadas que permite um ajuste linear por mínimos quadrados, se considerarmos as premissas estatísticas desse.

Para que se possa obter parâmetros de intercepto e inclinação a partir de uma regressão linear, é necessário que se cumpra as *premissas estatísticas* de 1) distribuição normal de resíduos, 2) homogeneidade de variâncias, e 3) independência das variáveis. Se observarmos as três linearizações, tanto a de Eadie-Hofstee como a de Hanes-Woolf não cumprem a premissa de independência, já que a variável dependente (y) é função da independente (x).

Para que uma transformação por duplos-recíprocos possa ser utilizada mais fielmente à obtenção de parâmetros cinéticos, contudo, pode-se adotar o cômputo de peso na fórmula de ajuste linear, tal como sugerido por Wilkinson (Wilkinson 1961), considerando-o como o recíproco das variâncias estimadas. Nesse caso, o ajuste linear considerando o quadrado do vetor de erros aleatórios como variância e o peso como seu recíproco ($1/s^2$), pode ser esboçado como:

```
# Regressão linear ponderada de Lineweaver-Burk

S <- seq(0.1, 1, length.out = 20)
Vm <- 10
Km <- 0.5
set.seed(1500)
erro <- runif(20, 0, 1)
v <- Vm * S / (Km + S) + erro
inv.S <- 1 / S
inv.v <- 1 / v
reg.LB.peso <- lm(inv.v ~ inv.S, weights = 1 / erro^2) # expressão para
 # ajuste linear
summary(reg.LB.peso) # resultados do ajuste
```

```
Call:
lm(formula = inv.v ~ inv.S, weights = 1/erro^2)

Weighted Residuals:
```

```
     Min       1Q   Median       3Q      Max
-0.04779 -0.02231 -0.01849  0.00162  0.04830

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.117327   0.002545   46.11  < 2e-16 ***
inv.S       0.034906   0.001452   24.04 3.93e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.02967 on 18 degrees of freedom
Multiple R-squared:  0.9698,    Adjusted R-squared:  0.9681
F-statistic:   578 on 1 and 18 DF,  p-value: 3.932e-15
```

Ainda que os ajustes tenham sido realizados para valores com baixa variabilidade de erros uniformes, uma comparação entre esse resultado e o da Figura 5.6 revela ligeira superioridade para o primeiro, com maior valor para o coeficiente de determinação R^2, menor para *p-valor*, e menores para o *erro-padrão* das estimativas.

Complementarmente, pode-se obter uma comparação estatística entre o modelo linear simples e o que adotou o peso estatístico por:

```
anova(reg.LB, reg.LB.peso)
```

```
Analysis of Variance Table

Model 1: inv.v ~ inv.S
Model 2: inv.v ~ inv.S
  Res.Df       RSS Df Sum of Sq F Pr(>F)
1     18 0.0038295
2     18 0.0158493  0  -0.01202
```

Sec 5.3 Ajuste não-linear

Ainda que linearizações sejam frequentemente utilizadas mesmo hoje em dia, principalmente para discernir entre modelos cinéticos distintos, a determinação precisa dos parâmetros de catálise é contudo melhor conduzida por ajuste ou regressão não-linear. Esse ajuste tem por objetivo a determinação de parâmetros de uma equação (*Vm* e *Km*, no caso) sem a necessidade de qualquer transformação dos dados, eliminando por essa razão os erros associados.

O ajuste não-linear difere do linear em algumas características, tais como:

1. A busca iterativa de um valor mínimo (local ou global) para a soma dos quadrados dos erros das estimativas;
2. a necessidade de um valor inicial para os parâmetros (sementes);
3. a linearidade nos erros e no gradiente da função sobre os parâmetros.
4. a necessidade de algoritmo mais sofisticado para solução simbólica e matricial para minimizar a derivada da função sobre cada parâmetro;
5. a necessidade de programa que trabalhe com álgebra matricial (computador, dispositivo móvel ou calculadora);
6. o uso de algoritmos mais sofisticados (Gauss, Newton-Raphson, Levenberg-Marquadt, Simplex).

7. o emprego da equação original do modelo, por vezes de difícil linearização.

5.3.1 Ajuste não-linear da equação de Michaelis-Mentem

Para uma regressão não-linear da equação de Michaelis-Mentem reproduzindo-se a simulação exemplificada na Figura 5.3:

```
# Regressão não linear para simulação de eq. de MM

Vm <- 10
Km <- 0.5
set.seed(1500)
erro <- runif(20, 0, 1)
```

```
S <- seq(0, 1, length.out = 20)
v <- Vm * S / (Km + S) + erro
dat.Sv <- data.frame(S, v) # criação de planilha com S e v
plot(v ~ S,
  type = "p", from = 0, to = 1, n = 20,
  xlab = "[S}", ylab = "v"
) # construção do gráfico de MM

nl.MM <- nls(v ~ Vm * S / (Km + S), start = list(Vm = 7, Km = 0.2),
             data = dat.Sv) # linha de código para ajuste não linear
lines(S, fitted(nl.MM), col = "red") # sobreposição da linha
```

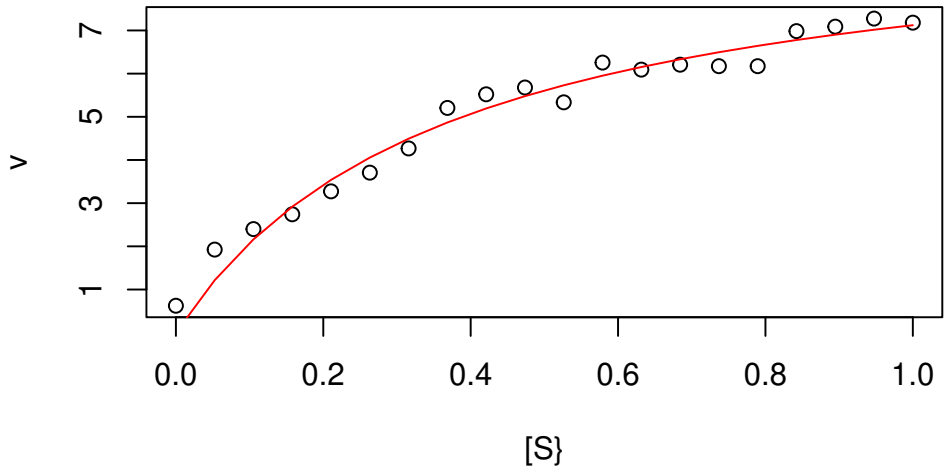


Figura 5.12: Ajuste não linear para a simulação experimental da equação de Michaelis-Menten.

```
 # ajustada
summary(nl.MM) # sumário dos resultados
```

```
Formula: v ~ Vm * S/(Km + S)

Parameters:
   Estimate Std. Error t value Pr(>|t|)
Vm  9.75490    0.52114  18.718 3.01e-13 ***
Km  0.36979    0.05015   7.373 7.68e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3514 on 18 degrees of freedom

Number of iterations to convergence: 5
Achieved convergence tolerance: 1.64e-06
```

Observe agora pela tabela de sumário do ajuste que os parâmetros são fornecidos diretamente, sem necessidade de transformação, como no ajuste linear. A tabela diferencia-se do sumário de ajuste linear por apresentar o número de iterações para convergência, e o valor de convergência para tolerância. Contudo, não apresenta o coeficiente de determinação R^2, mas somente o erro padrão residual. Ainda que a discussão esteja longe da proposta desta obra, alguns autores declinam do emprego de R^2 em regressão não-linear pelo mesmo ser decorrente de relações lineares entre os parâmetros, o que não ocorre no caso.

Assim como para ajuste linear, os parâmetros que abrangem a função `nls` envolvem:

```
args(nls)
```

```
function (formula, data = parent.frame(), start, control = nls.control(),
    algorithm = c("default", "plinear", "port"), trace = FALSE,
    subset, weights, na.action, model = FALSE, lower = -Inf,
    upper = Inf, ...)
NULL
```

Além do pacote incluido na distribuição básica do `R` e que permite ajustes não-lineares (`stats`), existem diversos outros que permitem ajustes com algoritmos, avaliações e plotagens variadas, tais como `nlme` (*mixed-effects*), `nlrwr`, `nlstools`, `nls2`, `nls.multstart`, `minpack.lm` (algoritmo de Levenberg-Marquadt), `nlshelper`, e `nlsLM`.

5.3.2 Algumas vantagens do modelo linear sobre o não-linear

Ainda que a estimativa de parâmetros de modelos não lineares seja mais precisa utilizando-se ajustes também não lineares, o algoritmo linear oferece algumas vantagems, entre as quais:

1. É mais fácil, com algoritmo simplificado, e mesmo pelo uso de somatórias de algumas quantidades envolvendo x e y, sendo resolvido com calculadora científica simples, ou mesmo à mão;
2. é mais intuitivo visualmente, posto que o modelo final será sempre uma reta;
3. possui apenas dois parâmetros na equação, intercepto e inclinação;
4. requer poucas medidas, já que uma reta se constroi com apenas dois pontos;
5. não requer sementes para estimativas iniciais o que, a depender do modelo não-linear, pode ser bem abstrato, culminando em mínimos locais ou mesmo na falta de solução para o ajuste;
6. permite interpretação experimental quando há fuga da linearidade;
7. independe de um modelo físico específico;
8. não requer, por vezes, a necessidade de constrição de resultados (*constraints*), por exemplo instruindo o algoritmo a buscar uma estimativa obrigatoriamente de valor positivo para o parâmetro.
9. relações lineares e transformações são encontradas em inúmeros modelos físicos nas Ciências Naturais.

Sec 5.4 Enzimas alostéricas

A *alosteria* constitui um dos principais recursos da metabolismo para a regulação dos níveis de compostos celulares. De etiologia grega (*allos = outro, stereos = estrutura*), uma *enzima alostérica* é aquela que altera seu perfil catalítico em função de transições conformacionais mediadas por moléculas que interagem fora de seu sítio ativo, sejam elas substrato, coenzimas, ou outros compostos (Traut 2007). O efeito resultante constitui em uma modulação da atividade enzimática, quer ativando-a ou inibindo-a. Enzimas alostérica comportam-se portanto como *enzimas regulatórias* em uma rota metabólica, e cuja atividade pode ser modulada em função de *retroinibição* ou *inibição por feedback*, bem como por *ativação pelo precursor* (Leone 2021).

A equação que define uma enzima alostérica em função do teor de seu substrato dada abaixo:

$$v = \frac{Vm * S^n}{(Km^n + S^n)} \tag{5.8}$$

Onde nH representa o *coeficiente de cooperatividade ou constante de Hill* para a ligação com moléculas de S (de maneira similar à ligação de O_2 à hemoglobina. De modo geral, o valor de nH pode ser inferior à unidade (*cooperatividade negativa*) ou superior a essa (*cooperatividade positiva*). Para ilustrar o comportamento cinético de uma enzima alostérica, segue o trecho abaixo, que também introduz outro formato para representar curvas no `R` nomeando a variável independente (x).

```
# Gráfico para enzima alostérica

v <- function(S, Vm = 10, Km = 3, nH = 2) {
  Vm * S^nH / (Km^nH + S^nH)
}
curve(v,
  from = 0, to = 10, n = 100, xlab = "S", ylab = "v",
```

```
  bty = "L"
) # eixos em L
```

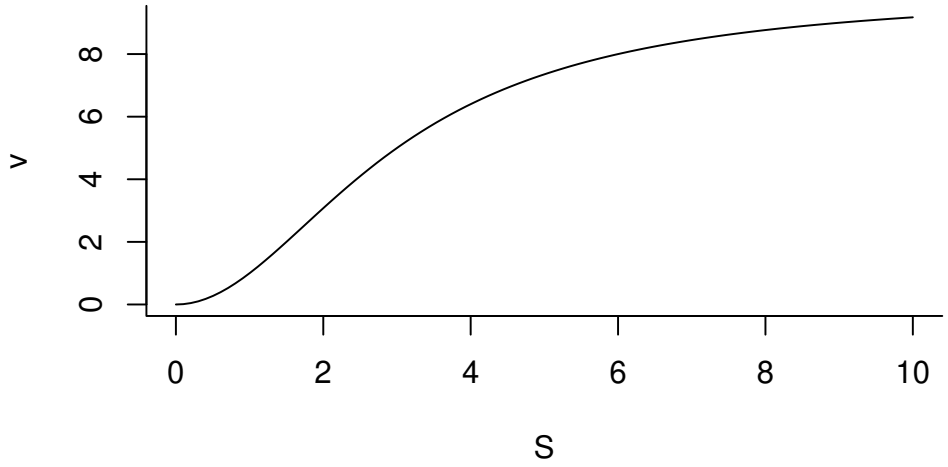


Figura 5.13: Gráfico do Michaelis-Mentem para uma enzima alostérica

Interessante também observar como o perfil enzimático alostérico se define frente à variação do coeficiente nH.

```
# Influência da constante de Hill (nH) sobre uma enzima alostérica

nH <- seq(from = 0.1, to = 3, length.out = 7) # sequência para 7 valores de nH
for (i in 1:length(nH)) { # loop para adicionar curva alostérica a cada valor de nH
  add <- if (i == 1) FALSE else TRUE # controle de fluxo
  v <- function(S, Vm = 10, Km = 3, a = nH[i]) {
    Vm * S^a / (Km^a + S^a)
  }
  curve(v,
    from = 0, to = 4, n = 500, col = i, xlab = "S", ylab = "v",
    bty = "L", add = add
  )
}
arrows(0, 5, 3, 2, length = 0.1, angle = 45, col = "blue") # seta para nH
text(0.5, 5.2, "nH", col = "blue") # indexador para nH
```

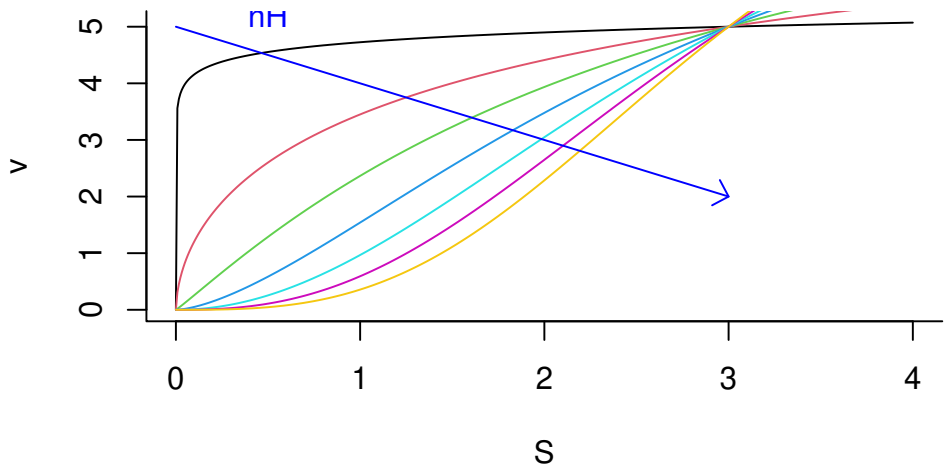


Figura 5.14: Perfil alostérico de uma enzima sob variação do coeficiente de cooperatividade nH.

Capítulo vasto na enzimologia e com aplicação direta em Química, Farmacologia, Biotecnologia, Biomedicina e áreas afins, a inibição enzimática encontra-se no âmago dos fármacos, medicamentos e biosensores. Sob um ponto de vista simplificado, a atividade enzimática pode ser reduzida na presença de vários efetores, entre moléculas endógenas ou exógenas do metabolismo celular, incluindo o próprio substrato. Genericamente a *inibição enzimática* classifica-se como *irreversível* quando a ativide decai pela ligação covalente de um inibidor, ou *reversível*, quando há um equilíbrio de associação/dissociação com a macromolécula. A seguir serão ilustradas inibições reversíveis.

6.0.1 Inibição pelo substrato

Uma inibição enzimática comum ao metabolismo é a protagonizada pelo próprio substrato em excesso no meio, sendo definida por:

$$v = \frac{Vm * S}{S(1 + \frac{S}{Ks}) + Km} \tag{6.1}$$

Dessa forma, o gráfico resultante de uma inibição por excesso de substrato pode ser reproduzido por:

```
# Inibição por excesso de S

S <- seq(0, 10, 0.1)
v_alos <- function(S, Vm = 10, Km = 0.5, Ks = 2) {
  Vm * S / (S * (1 + S / Ks) + Km)
}
curve(v_alos, xlim = c(0, 10), xlab = "S", ylab = "v")
```

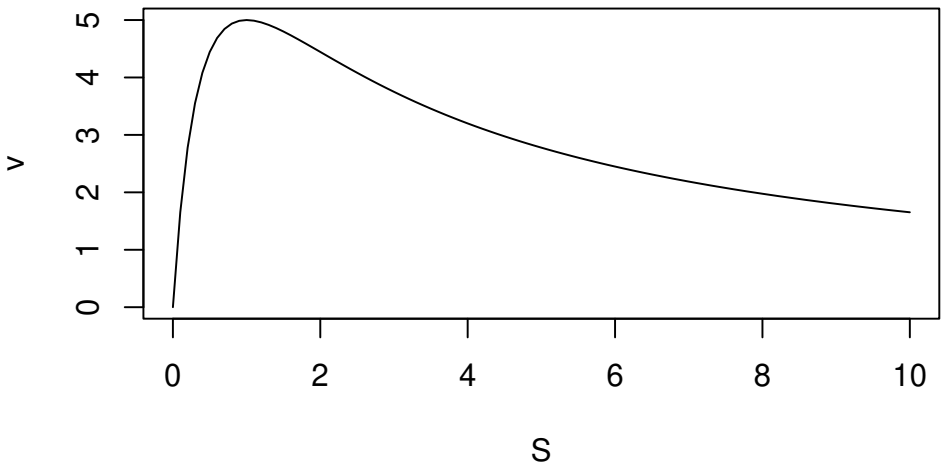


Figura 6.1: Comportamento cinético de uma enzima inibida por excesso de substrato.

Percebe-se pela Figura 6.1 que a atividade da enzima alcança um limite sendo reduzida com o aumento do teor de substrato.

6.0.2 Modelos de inibição enzimática

A inibição de enzimas por moléculas que não o próprio substrato pode ser representada por um diagrama no qual o efetor altera um dos elementos representados na Equação 5.1, tal como na figura abaixo.

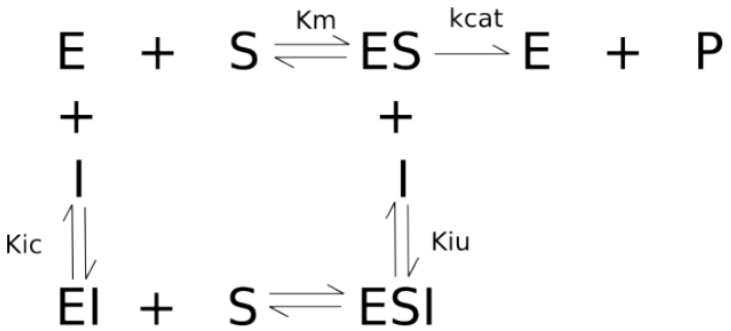


Figura 6.2: Diagrama representativo dos tipos de inibição enzimática. kcat: constante catalítica; Ki: constante de equilíbrio de dissociação do inibidor, com índices para inibição competitiva (Ki), incompetitiva (Kiu) e não competitiva (Kic e Kiu).

Nesse caso, pode-se definir os três tipos principais de inibição enzimática reversível como *inibição competitiva, incompetitiva e não competitiva (pura ou mista)*. Em síntese, a *inibição competitiva* dá-se quando o inibidor complexa-se ao sítio ativo da enzima; a *inibição incompetitiva*, quando o inibidor interage com o complexo enzima-substrato; e a *inibição não competitiva*, quando o inibidor liga-se em outro local que não o sítio ativo da enzima, podendo ainda ser *pura* (Kiu = Kic) ou *mista* (Kiu > ou < Kic).

Um modelo matemático que abrange esses três tipos de inibição enzimática é descrito na equação abaixo:

$$v = \frac{Vm * S}{Km(1 + \frac{I}{Kic}) + S(1 + \frac{I}{Kiu})} \tag{6.2}$$

Dessa forma, a Equação 6.2 reduz-se em seus termos multiplicadores no denominador, em função do tipo de inibição enzimática presente, até o modelo primitivo de Michaelis-Mentem, quando na ausência do inibidor.

6.0.2.1 Curva de Michaelis-Mentem para modelos de inibição enzimática

Podemos simular no `R` as cuvas michaelianas para modelos clássicos de inibição, considerando valores para as constantes de equilíbrio de dissociação dos inibidores como Kic = 0.2, e Kiu = 1, como no trecho de código abaixo.

```
# Inibição clássica & Michaelis-Mentem

par(mfrow = c(2, 2)) # divide a área de plotagem
S <- seq(0, 10, 0.1) # geração de teores de S
contr <- function(S, Vm = 10, Km = 0.5) {
  Vm * S / (Km + S)
} # função de MM, sem inibição
curve(contr, xlim = c(0, 10), xlab = "S", ylab = "v", main = "Competitiva")
 # cuva controle; veja que o título tem que ser adicionado para o 1a. de par
 # de curvas, controle e inibição

# Modelos de inibição:
```

```r
# Competitiva
comp.i <- function(S, Vm = 10, Km = 0.5, I = 2, Kic = 0.2) {
  Vm * S / (Km * (1 + I / Kic) + S)
}
curve(comp.i, add = TRUE, col = "red", lty = 2) # competitiva

# Não competitiva pura
pura.i <- function(S, Vm = 10, Km = 0.5, I = 2, Ki = 1) {
  Vm * S / (Km * (1 + I / Ki) + S * (1 + I / Ki))
}
curve(contr, xlim = c(0, 10), xlab = "S", ylab = "v", main = "Não Compet. Pura")
curve(pura.i, add = TRUE, col = "red", lty = 2) # não competitiva pura (Kiu=Kic=Ki)

# Não competitiva mista
mista.i <- function(S, Vm = 10, Km = 0.5, I = 2, Kic = 0.2, Kiu = 1) {
  Vm * S / (Km * (1 + I / Kic) + S * (1 + I / Kiu))
}
curve(contr, xlim = c(0, 10), xlab = "S", ylab = "v", main = "Não Compet. Mista")
curve(mista.i, add = TRUE, col = "red", lty = 2) # não competitiva mista

# Incompetitiva
incomp.i <- function(S, Vm = 10, Km = 0.5, I = 2, Kiu = 1) {
  Vm * S / (Km + S * (1 + I / Kiu))
}
curve(contr, xlim = c(0, 10), xlab = "S", ylab = "v", main = "Incompetitiva")
curve(incomp.i, add = TRUE, col = "red", lty = 2) # incompetitiva
layout(1) # retorna à janela gráfica original
```

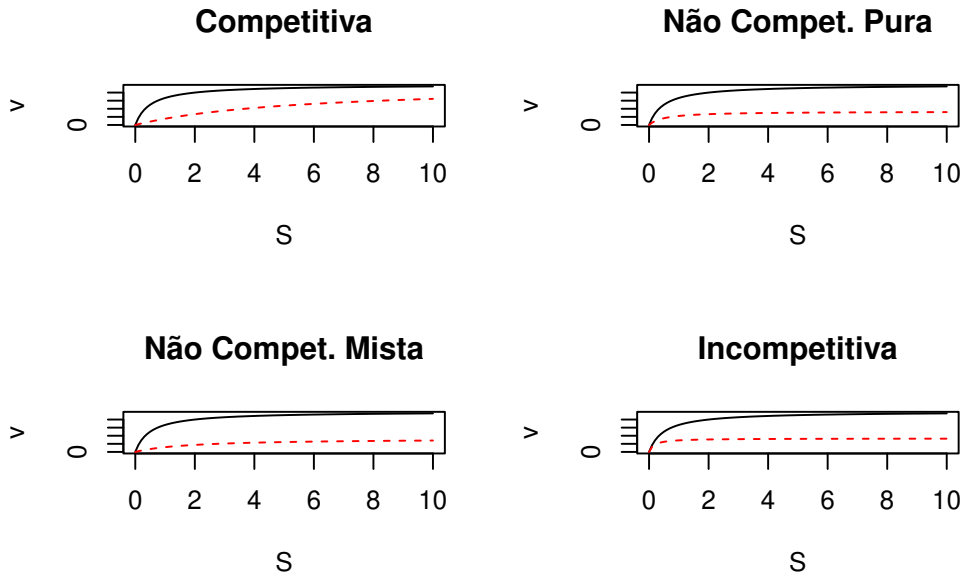


Figura 6.3: Curva de Michaelis-Mentem na presença de inibidores de comportamento clássico.

Perceba que para o *modelo competitivo* a velocidade limite *Vm* da reação tende a ser alcançada, já que a ocupação do sítio ativo da enzima é mutualmente exclusiva entre substrato e inibidor, privilegiando o primeiro quando em alto teor. Por outro lado, o valor de *Vm* não é tangível para os demais modelos de inibição, já que o inibidor se liga em outro sítio na enzima (*não competitivo*) ou ao próprio substrato (*incompetitivo*). Para visualizar alterações nos gráficos, experimente modificar os parâmetros da simulação (Vm, Km, Ki, Kic, Kiu, I).

Ainda que seja possível um discernimento do modelo competitivo dos demais, perceba também que isso só foi

possível por uma simulação que empregou um teor S 20 vezes maior que o valor de Km da reação. Isso nem sempre é possível na prática, como elencado abaixo, já que o emprego de altos teores de S:

1. Agrega maior custo financeiro ao ensaio.
2. Pode resultar em inibição por excesso de substrato.
3. Pode elevar a viscosidade do meio, reduzindo a taxa catalítica.

Percebe-se, da Figura 6.3 e das observações acima, a dificuldade em classificar o tipo de inibição enzimática baseado na observação direta de uma curva de Michaelis-Mentem.

6.0.2.2 Diagnóstico de modelos de inibição enzimática por Lineweaver-Burk

As transformações lineares da equação de Michaelis-Mentem são muito úteis no diagnóstico visual de modelos de inibição. Nesse sentido, o emprego da linearização por duplos-recíprocos para esses modelos resutará nas equações de inibição que seguem:

$$\frac{1}{v} = \frac{1}{Vm} + \frac{Km(1+\frac{I}{Kic})}{Vm} * \frac{1}{S} \quad ; competitivo \tag{6.3}$$

$$\frac{1}{v} = \frac{1}{Vm} + \frac{Km(1+\frac{I}{Ki})}{Vm} * \frac{1}{S(1+\frac{I}{Ki})} \quad ; puro \tag{6.4}$$

$$\frac{1}{v} = \frac{1}{Vm} + \frac{Km(1+\frac{I}{Kic})}{Vm} * \frac{1}{S(1+\frac{I}{Kiu})} \quad ; misto \tag{6.5}$$

$$\frac{1}{v} = \frac{1}{Vm} + \frac{Km}{Vm} * \frac{1}{S(1+\frac{I}{Kiu})} \quad ; incompetitivo \tag{6.6}$$

Observe que os termos multiplicadores inseridos em S e Km na equação de duplo-recíproco apenas alteram seu formalismo apresentado Equação 5.4. Dessa forma, os modelos de inibição enzimática podem ser ilustrados pelo `R` junto à transformação de Lineweaver-Burk (ou qualquer outra), como abaixo.

```
# Diagnóstico de inibição por Lineweaver-Burk


# Substrato e Inibidor
S <- seq(0.1, 10, length = 10) # cria um vetor para substrato
I <- 2 # concentração de inibidor

# Parâmetros cinéticos:
Km <- 0.5
Vm <- 10
Kic <- 0.2
Ki <- 0.2
Kiu <- 1

# Equações
v <- Vm * S / (Km + S) # equação de MM
v.comp <- Vm * S / (Km * (1 + I / Kic) + S) # competitivo
v.puro <- Vm * S / (Km * (1 + I / Ki) + S * (1 + I / Ki))
 # não competitivo puro
v.misto <- Vm * S / (Km * (1 + I / Kic) + S * (1 + I / Kiu))
 # não competitivo misto
v.incomp <- Vm * S / (Km + S * (1 + I / Kiu))

# Gráficos
par(mfrow = c(2, 2)) # área de plot pra 4 gráficos

plot(1 / S, 1 / v, type = "l", main = "Competitivo", ylim = c(0, 2))
points(1 / S, 1 / v.comp, type = "l", col = "red")
```

```
plot(1 / S, 1 / v, type = "l", main = "Puro", ylim = c(0, 5))
points(1 / S, 1 / v.puro, type = "l", col = "red")
plot(1 / S, 1 / v, type = "l", main = "Misto", ylim = c(0, 2))
points(1 / S, 1 / v.misto, type = "l", col = "red")
plot(1 / S, 1 / v, type = "l", main = "Incompetitivo", ylim = c(0, 1))
points(1 / S, 1 / v.incomp, type = "l", col = "red")
```

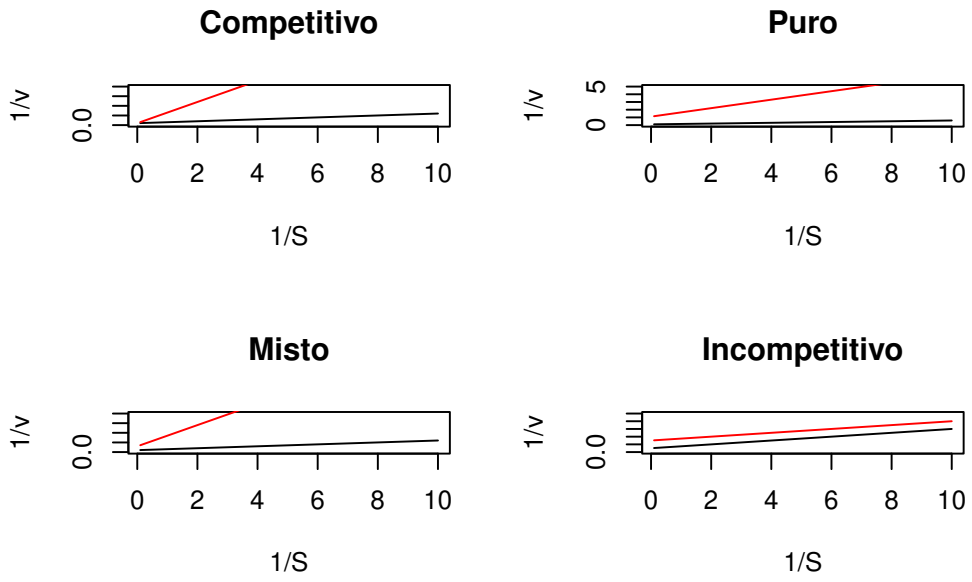


Figura 6.4: Diagnóstico de modelos de inibição enzimática por Lineweaver-Burk.

```
layout(1) # volta à janela gráfica normal
```

Agora a distinção de modelos de inibição se torna mais evidente pela linearização. Assim como mencionado para os modelos representados na equação direta de Michaelis-Mentem, pode-se variar os parâmetros cinéticos e experimentar a visualização dos duplos-recíprocos.

Novamente, ainda que a linearização permita um melhor diagnóstico do tipo de inibição presente, o ajuste não linear é mais adequado para a determinação das constantes de inibição (*Ki*'s), uma vez que não agrega os erros advindos das transformações lineares (embora a inserção de pesos estatísticos possa aliviar a imprecisão dos resultados).

6.0.2.3 Ki & IC_{50}

A concentração inibitória a 50% do teor de inibidor, definida como IC_{50}, pode ser determinada empiricamente sem o conhecimento dos parâmetros de catálise enzimática envolvidos. Para isso, basta se obter um valor de inibição relativa num ensaio a concentração fixa de S, variando-se o teor de inibidor. De fato, análogos ao IC_{50} existem em ampla gama nas Ciências Naturais, não envolvendo necessariamente qualquer informação cinética ou termodinâmica dos compostos envolvidos, mas tão somente a informação empírica do resultado. Exemplificando, os parâmetros DE_{50} (dose efetiva) ou DL_{50} (dose letal), e mesmo projeções de X_{50}, tal como Tm (temperatura de desnaturação a 50%), e o valor de pKa em tampões (pH em que as espécies encontram-se 50% ionizadas/protonadas em solução).

No entanto, existe uma relação útil entre a constante de equilíbrio de dissociação do inibidor Ki e o valor de IC_{50} que permite sua permuta, desde que conhecido o modelo de inibição (Yung-Chi e Prusoff 1973). Generalizando para os modelos de inibição, pode-se definir uma equação geral pra relação de Cheng-Prusoff como:

$$IC_{50} = \frac{(1 + \frac{S}{Km})}{(\frac{1}{Kic}) + (\frac{1}{Km * Kiu})} \tag{6.7}$$

Exemplificando, para um *modelo competitivo* de inibição, onde Kiu é nulo:

$$IC_{50} = Kic(1 + \frac{S}{Km}) \tag{6.8}$$

Como acima mencionado, o valor de IC_{50} pode ser obtido a partir de dados experimentais de inibição relativa (v/Vm, por ex) em diferentes concentrações de inibidor fixando um valor de S. Nesse caso, podemos ilustrar no `R` a obtenção de IC_{50}, utilizando-se um ajuste não linear para a equação de quatro parâmetros que segue (*curva de Rodbard*, DeLean, Munson, e Rodbard (1978)).

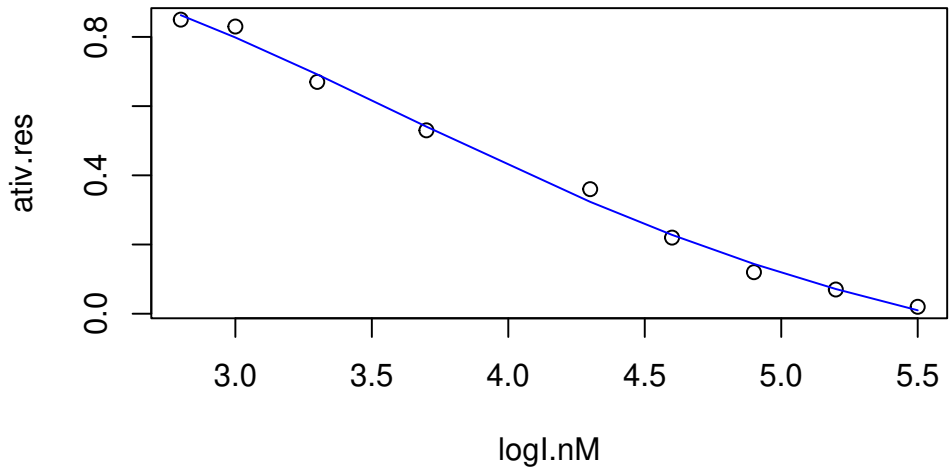
$$ativ.residual\,\% = \frac{v}{Vm} = inf + \frac{sup - inf}{1 + log(\frac{I}{IC_{50}})^{nH}})$$

{#eq:eqRodb}

```
# Ajuste não-linear para curva de IC50

logI.nM <- c(5.5, 5.2, 4.9, 4.6, 4.3, 3.7, 3.3, 3, 2.8)
# conc. de I, em unidade log10
ativ.res <- c(0.02, 0.07, 0.12, 0.22, 0.36, 0.53, 0.67, 0.83, 0.85)
# ativ. residual, v/Vm
dados <- data.frame(logI.nM, ativ.res) # criação do dataframe
plot(ativ.res ~ logI.nM, dados) # plot dos dados
ic50.fit <- nls(formula(ativ.res ~ inf + (sup - inf) /
                          (1 + (logI.nM / logIC50)^nH)),
                algorithm = "port", data = dados,
                start = list(inf = 0, sup = 0.80, logIC50 = 4, nH = 10),
                lower = c(inf = -Inf, sup = -Inf,
                          logIC50 = 0, nH = -Inf)) # ajuste não linear
summary(ic50.fit) # sumário do ajuste
```

```
Formula: ativ.res ~ inf + (sup - inf)/(1 + (logI.nM/logIC50)^nH)

Parameters:
        Estimate Std. Error t value Pr(>|t|)
inf      -0.3211     0.2932  -1.095  0.32348
sup       1.1200     0.2311   4.847  0.00469 **
logIC50   4.0807     0.2309  17.675 1.06e-05 ***
nH        4.0540     1.7462   2.322  0.06792 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.02769 on 5 degrees of freedom

Algorithm "port", convergence message: relative convergence (4)
```

```
lines(logI.nM, fitted(ic50.fit), col = "blue") # linha ajustada
```




```
# E para extrair o valor de IC50...
IC50 <- 10^(coef(ic50.fit)[3]) # extração do 3o. parâmetro da tabela
 # de ajuste, isto é: logIC50:
IC50
```

```
 logIC50
12042.04
```

Perceba que o parâmetro de *logIC50* foi extraído da tabela de ajuste não linear pelo comando `coef`. Isto é muito útil quando desejamos utilizar um coeficiente obtido em cálculos automáticos (programáveis), como veremos mais adiante. Por ora, faz-se interessante apresentar o parâmetro de *IC50* obtido de forma mais elegante.

Para isso, podemos utilizar duas funções do `R` para exprimir resultados quantitativos junto à caracteres (palavras, frases): `print()` e `cat`. O trecho de código abaixo ilustra esse *output*, e algumas diferenças.

```
cat("Valor de IC50 (nM):", IC50, "\n")
```

```
Valor de IC50 (nM): 12042.04
```

```
print(paste("Valor de IC50 (nM):", IC50))
```

```
[1] "Valor de IC50 (nM): 12042.0403466162"
```

Basicamente, `print` exibe aspas e indexa o nome da coluna, enquanto `cat` os omite. Em adição, pode-se perceber outra variação no formato de impressão entre os dois comandos pelo exemplo abaixo:

```
print(paste("teores:", c(10, 25, 50)))
```

```
[1] "teores: 10" "teores: 25" "teores: 50"
```

```
cat("teores:", c(10, 25, 50))
```

```
teores: 10 25 50
```

Outra possibilidade no `R` é a de se reduzir o número de casas decimais apresentados. Nesse caso, pode-se utilizar o comando `round`.

```
IC50 <- 10^(coef(ic50.fit)[3])
print(paste("Valor de IC50 (nM):", round(IC50, digits = 2)))
```

```
[1] "Valor de IC50 (nM): 12042.04"
```

```
# arredondamento para duas casas decimais
```

Mais uma vez, salienta-se a existência de alguns pacotes úteis do `R` para o cálculo de *IC50*, entre esse o pacote `drc` (*dose-response curve*).

Sec 6.1 Diagnóstico estatístico de inibição enzimática

Em paralelo à inspeção visual dos gráficos de linearização para inibição enzimática, é possível validar-se um modelo sobre outro por uma análise de dispersão de erros dos modelos. Mas também é possível o emprego da função `BIC` ou da função `AIC` do `R`, e que respectivamente calculam valores para o *Critério de Informação Bayseiano* (Spiess e Neumeyer 2010) ou do *Critério de Informação de Akaike* (Akaike 1974). Em comum esse parâmetros calculam um valor relativo de informação não computada por um modelo avaliado. O menor valor encontrado para ambos espelha a solução do melhor modelo de ajuste.

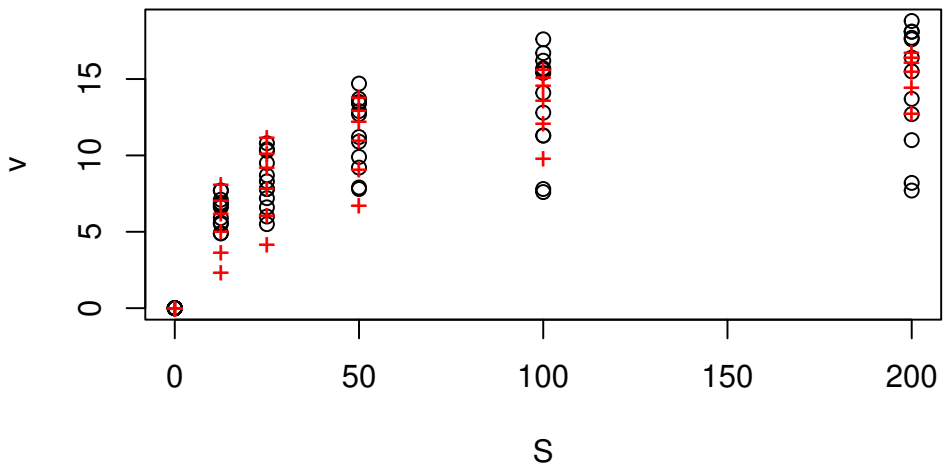
Matematicamente, *BIC* e *AIC* podem ser expressos como:

$$BIC = p * ln(n) - 2 * ln(RSE) AIC = n * ln(\frac{RSE}{n}) + 2k + [\frac{2k(k+1)}{n-k-1}]) \tag{6.9}$$

Onde p representa o no. de parâmetros do modelo, n o número total de pontos experimentais, k o fator $p+1$, e RSE o valor da soma dos quadrados dos resíduos (*residual sum squares*).

Para exemplificar o uso desses parâmetros de qualidade do modelo estatístico, pode-se empregar um conjunto de dados contido no pacote `nlstools`, provendo o ajuste, plotagem, inspeção de resíduos, e aplicação de *BIC* e *AIC*:

```
# Aplicação de critérios de informação para ajuste de curvas cinéticas

library(nlstools)
comp <- nls(compet_mich, vmkmki, list(Km = 1, Vmax = 20, Ki = 0.5))
# ajuste competitivo, com dados, equação e sementes fornecidas
# pelo pacote nlstools
plotfit(comp, variable = 1) # comando de plotagem do pacote
```

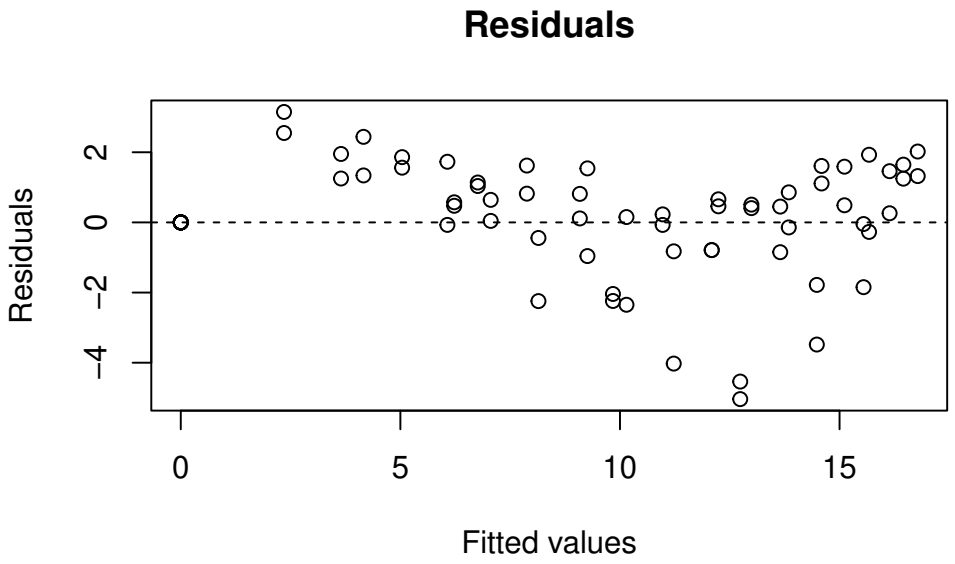



```
summary(comp)
```

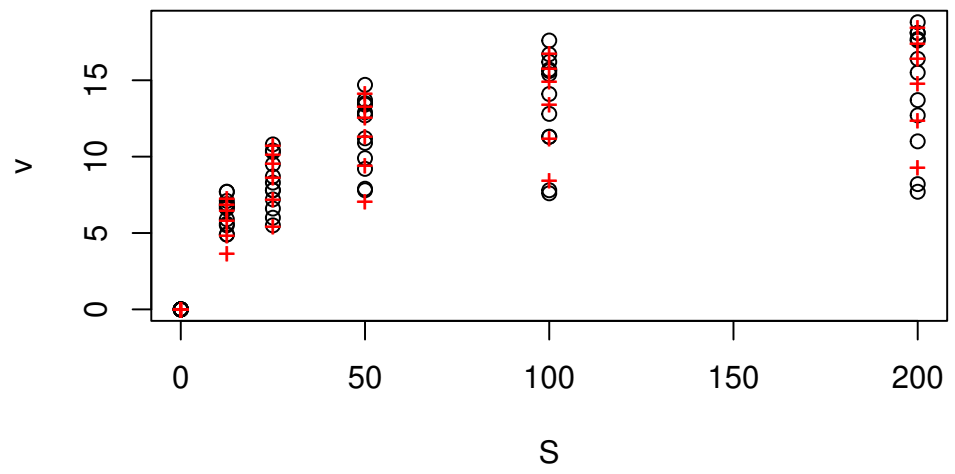
```
Formula: v ~ S/(S + Km * (1 + I/Ki)) * Vmax

Parameters:
     Estimate Std. Error t value Pr(>|t|)
Km    15.2145     2.5005   6.085 5.79e-08 ***
Vmax  18.0557     0.6288  28.713  < 2e-16 ***
```

```
Ki     22.2822     4.9060   4.542 2.30e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.603 on 69 degrees of freedom

Number of iterations to convergence: 11
Achieved convergence tolerance: 5.116e-06
```

```
res_comp <- nlsResiduals(comp) # resíduos do ajuste
plot(res_comp, which = 1) # plotagem de resíduos
```



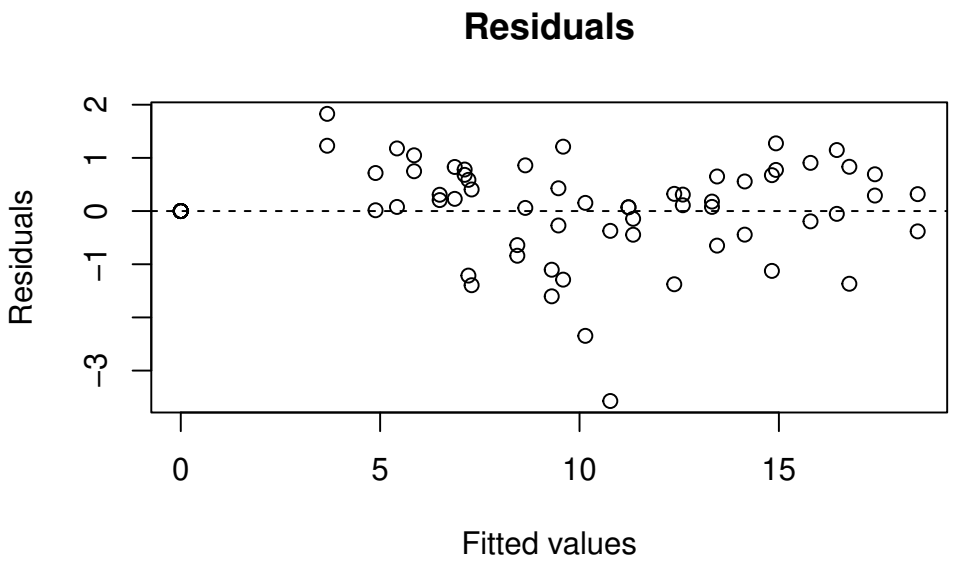

```
noncomp <- nls(non_compet_mich, vmkmki, list(Km = 1, Vmax = 20, Ki = 0.5))
# o mesmo que acima, mas para o modelo não competitivo
plotfit(noncomp, variable = 1)
```



```
summary(noncomp)
```

```
Formula: v ~ S/((S + Km) * (1 + I/Ki)) * Vmax

Parameters:
     Estimate Std. Error t value Pr(>|t|)
Km    22.7787     1.4738   15.46   <2e-16 ***
Vmax  20.5867     0.4306   47.80   <2e-16 ***
Ki   101.3563     7.3303   13.83   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8925 on 69 degrees of freedom

Number of iterations to convergence: 7
Achieved convergence tolerance: 8.27e-06
```

```
res_noncomp <- nlsResiduals(noncomp)
plot(res_noncomp, which = 1)
```




```
BIC(comp, noncomp) # Critério de informação de Baysean
```

```
        df      BIC
comp     4 286.2994
noncomp  4 201.9981
```

```
AIC(comp, noncomp) # Critério de informação de Akaike
```

```
        df      AIC
comp     4 277.1928
noncomp  4 192.8915
```

Pode-se observar na comparação dos ajustes não lineares que o modelo não competitivo ajustou-se melhor que o modelo competitivo (valores menores para BIC e AIC)

Sec 6.2 Cinética de estado pré-estacionário

Existem basicamente três tipos de comportamentos cinéticos para as enzimas: comportamento de *Henry-Michaelis-Mentem-Briggs-Haldane*, sucintamente denominado por *michaeliano* ou de *estado estacionário (steady-*

state), comportamento de *fase lag* (quando o substrato leva algum tempo para ser convertido em produto), e comportamento de *burst*, *transiente*, ou de *estado pré-estacionário* (quando uma fase com rápida liberação de produto precede o estado estacionário). Algumas enzimas trabalham seguindo a cinética de *burst*, entre as quais algumas nucleosidades e glicosidases, e dehalogenases Tang et al. (2003).

A cinética de estado pré-estacionário segue um formalismo um pouco distinto, e que depende do quantitativo de etapas reacionais. Exemplificando abaixo para uma reação de 3 etapas (Johnson 1992):

$$E + S \underset{km1}{\overset{k1}{\rightleftharpoons}} E * S \underset{km2}{\overset{k2}{\rightleftharpoons}} E * P \overset{k3}{\rightarrow} E + P \tag{6.10}$$

Nesse caso, as equações derivadas das observações experimentais, e que conduzem à determinação das constantes de velocidade são:

$$kobs = k2 + km2 + k3 \tag{6.11}$$

$$Ao = \frac{k2 * (k2 + km2)}{kobs^2} \tag{6.12}$$

$$kcat = \frac{k2 * k3}{kobs} \tag{6.13}$$

Onde *kobs* e *Ao* representam parâmetros experimentais de constante de velocidade observada e amplitude, respectivamente. Esses parâmetros podem ser obtidos a partir do ajuste não linear da equação abaixo aos dados experimentais:

$$P = Ao(1 - e^{-kobs} + kcat * t) \tag{6.14}$$

O trecho de código que segue simula uma curva de comportamento pré-estacionário, quando conhecidas as constantes de velocidade que determinam os parâmetros experimentais.

```
# Curva de MM em enzima de comportamento pré-estacionário

# Parâmetros
k2 <- 387
km2 <- 3
k3 <- 22
xmin <- 0
xmax <- 0.075 # definição de limites para função

# Variáveis da equação de simulação (função dos parâmetros)
kobs <- k2 + km2 + k3
Ao <- k2 * (k2 + km2) / kobs^2
kcat <- k2 * k3 / kobs

# Definição da função de simulação
sim <- function(x, kobs, Ao, kcat) {
  Ao * (1 - exp(-kobs * x)) + kcat * x
}

# Curval de simulação
curve(sim(x, kobs = kobs, Ao = Ao, kcat = kcat),
  col = "blue",
  type = "o", xlim = c(xmin, xmax), cex = 0.5,
  xlab = "tempo", ylab = "[P]"
)
```

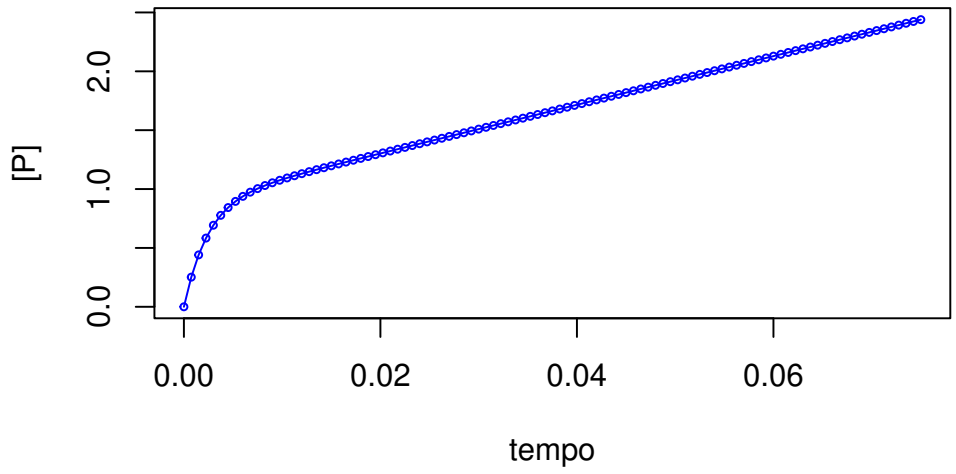


Figura 6.5: Formação de produto num modelo cinético de estado pré-estacionário.

Note pela Figura 6.5 que na cinética de estado transiente, existe uma fase pré-estacionária inicial eleva o teor de produto P rapidamente, e que antecede a fase estacionária de liberação constante de P.

Por outro lado, por vezes é necessário o oposto, ou seja, determinar as constantes de velocidade a partir do conhecimento dos parâmetros experimentais *kobs* e *Ao*. Nesse caso (e em tantos outros transdisciplinares) o `R` possui funções de minimização que permitem encontrar a raíz de equações lineares ou não lineares.

O procedimento envolve minimizar iterativamente um vetor de equações dadas as sementes para cada parâmetro. Para tal busca-se obter *f(x) = 0* pela diferença entre um valor de referência; ou seja, quando a solução encontrar x quando *f(x)-y = 0*. Exemplificando, supondo que f(x) seja *a+b/x*, e que y seja 3. Então a busca se dá no sentido de encontrar a e b em *a+b/x-3*.

Para a determinação das constantes de velocidade representadas na cinética transiente, vale mencionar a função `optim` em `stats` ou o pacote `rootSolve`, que buscam minimizar equações lineares e não lineares para encontrar os valores de seus parâmetros.

Na solução dos parâmetros para estado pré-estacionário, ilustra-se abaixo o emprego do `R` com `rootSolve`, adicionando ainda a busca para Km como segue.

$$Km = \frac{k3}{k2 + k3}$$

{#eq:burstKm}

```
# Cálculo de constantes cinéticas por solução de sistema de equações
# não lineares aplicadas à cinética de burst.

library(rootSolve)
kobs <- 0.06
Ao <- 50
kcat <- 300
Ks <- 15
# define os parâmetros de ajuste não linear obtidos por curva progressiva
# experimental, t x P;
# Obs: Ks obtido experimentalmente de curva de S x kobs

# Parâmetros
# x[1]=k2
# x[2]= k3
```

```
# x[3] = Km

# Modelo
model <- function(x) c(x[1] / kobs^2 - Ao, (x[1] * x[2]) / kobs - kcat,
                       Ks * x[2] / (x[1] + x[2]) - x[3])
# o modelo acima deve conter uma lista de equações cuja igualdade é zero,
# ou seja, f(x)=0
(ss <- multiroot(model, c(1, 1, 1))) # comando de execução do rootSolve
```

```
$root
[1]   0.18000 100.00000  14.97305

$f.root
[1]  0.000000e+00  1.136868e-13 -1.243054e-09

$iter
[1] 4

$estim.precis
[1] 4.143891e-10
```

```
# (sementes pro algoritmo)
```

Os resultados da minimização podem ser interpretados como:

1. *root* = valores de xi pra f(xi)=0 ; ou seja, *k2, k3, e Km*;
2. *f.root* = valor de cada função pra cada xi (deve ser próximo de zero para cada);
3. *iter* = no. iterações ;
4. *esti.precis* = estimativa da precisão.

A contemplar um capítulo ainda que extenso sobre cinética enzimática, existem inúmeros tópicos deixados de lado, dado o foco principal do emprego do `R` na solução de problemas quantitativos em biofísico-química. Dessa forma, omitimos diversos conceitos, tais como *cinética lenta de interação de substrato (slow binding), cinética de múltiplos substratos (reação sequencial e ping-pong), equação integrada de Michaelis-Menten e curvas progressivas, ativação de moduladores, influência de pH e temperatura na catálise, e enzimas multisítios*, entre vários.

Quando se menciona interação entre biomoléculas, normalmente se faz referência à processos adsortivos envolvendo um biopolímero (proteína, ácido nucleico, glicano), e um ligante de baixo peso molecular, embora o formalismo também se aplique com alguma restrição a interações entre biopolímeros, e mesmo células inteiras.

O formalismo mais comum para interação biomolecular é o que envolve a formação de complexo adsortivo entre uma proteína e um ligante (*ligand binding*), exemplificado para íons (Ca^{2+}, Mg^{2+}, etc), fármacos e candidatos, produtos naturais, e antígenos, dentre vários.

Perguntas simples acerca da *interação ligante-proteína* podem elucidar diversas características da formação de tais complexos, como:

1. Quanto de proteína/ligante estão presentes ?
2. Quanto do complexo é formado ?
3. Quão rápido o complexo associa/dissocia ?
4. Quais os mecanismos envolvidos ?

De modo geral, pode-se representar a interação ligante-proteína como segue:

$$P + L \underset{k_{off}}{\overset{k_{on}}{\rightleftharpoons}} PL \tag{7.1}$$

Onde P representa o teor de proteína livre, L o ligante livre, e PL o complexo formado. As taxas de reação são definidas para a formação (k_{on}; $M^{-1}s^{-1}$) e dissociação (k_{off}; s^{-1}) do complexo.

Dessa forma deduz-se a equação para a *isoterma de interação* do ligante com a proteína como segue:

$$Kd = \frac{[P] * [L]}{[P] + [L]} \tag{7.2}$$

Onde Kd representa a *constante de equilíbrio de dissociação* para o complexo PL formado, tal como condicionado ao equilíbrio de formação/dissociação do complexo ($v_{assoc} = v_{dissoc}$), e também definido como:

$$Kd = \frac{k_{off}}{k_{on}} \tag{7.3}$$

A partir da Equação 7.2 pode-se facilmente deduzir a expressão final para a interação de um ligante a um conjunto de sítios de mesma afinidade na proteína:

$$\nu = \frac{n * [L]}{Kd + [L]} \tag{7.4}$$

Sec 7.1 Modelos de Interação e Representações Lineares

Observe que a Equação 7.4 praticamente repete o formalismo já visto com a formação do complexo ativado de enzima-substrato (Equação 5.2), bem como sua representação resultante como uma hipérbole quadrática. De fato, ocorre essencialmente a substituição do parâmetro cinético v da reação pelo parâmetro termodinâmico ν ("nu", do Grego) para a isoterma de ligação. As demais quantidades envolvidas mantém-se análogas (P no lugar de E; L no lugar de S;Kd no lugar de Km; e n no lugar de $Vmax$).

Mantida essa similaridade com o formalismo da *equação de Michaelis-Menten*, da mesma maneira decorrem as linearizações para a Equação 7.4), bem como ajustes não lineares à mesma, na busca de uma *solução analítica* para

os parâmetros termodinâmicos *Kd* e *n*. Exemplificando um trecho de código para as linearizações mais comuns no tratamento de dados de interação ligante-proteína:

```
# Linearizações em interação bimolecular

L <- c(0.1, 0.2, 0.5, 1, 5, 10, 20) * 1e-6
Kd <- 1e-6
n <- 1
v <- n * L / (Kd + L)
par(mfrow = c(2, 3)) # estabelece área de plot pra 6 gráficos
plot(L, v, type = "o", main = "Direto")
plot(log(L), v, type = "o", main = "Langmuir")
plot(1 / L, 1 / v, type = "o", main = "Klotz")
plot(v, v / L, type = "o", main = "Scatchard")
plot(L, L / v, type = "o", main = "Woolf")
plot(log10(L), log10(v / (n - v)), type = "o", main = "Hill")
```

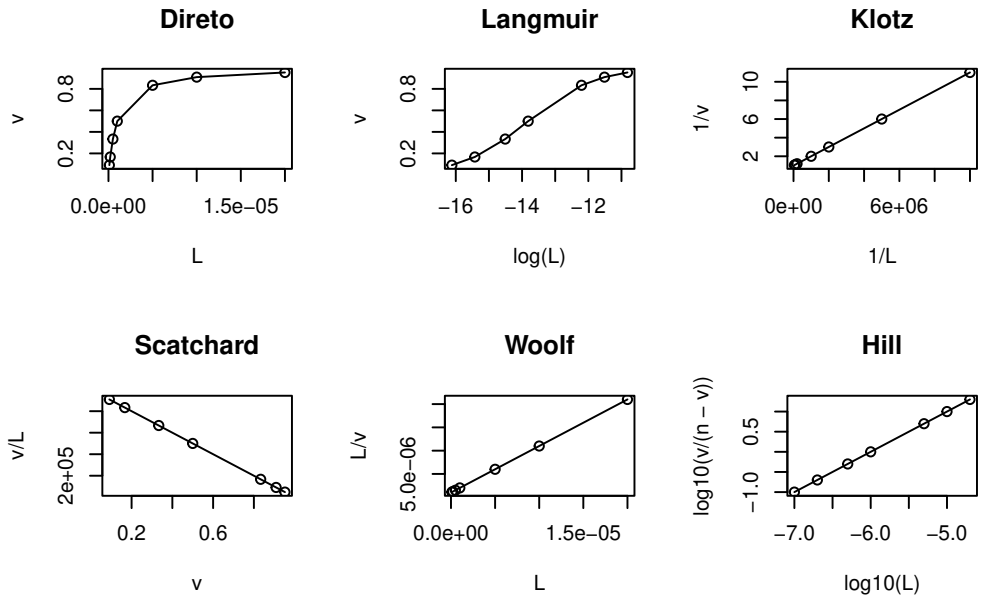


Figura 7.1: Principais linearizações da isoterma de ligação ligante-proteína.

```
par(mfrow = c(1, 1)) # volta à janela gráfica normal
```

Desvios da linearidade, por outro lado, são frequentemente utilizados como diagnósticos para modelos que distintos do de *homogeneidade de sítios de ligação* como acima (*heterogeneidade de sítios*, *criação de sítio*, *cooperatividade*. As equações abaixo descrevem esses modelos, e consideram K, *constante de equilíbrio de associação ligante-proteína*, como o reverso de *Kd*, a fim de tornar as expressões mais legíveis:

$$K = \frac{1}{Kd} \tag{7.5}$$

O *modelo de heterogeneidade de sítios* de ligação pressupõe que haja na proteína mais um sítio com afinidades distintas para o ligante (Dahlquist 1978). Formalmente esse modelo pode ser exemplificado para 2 conjuntos de sítios de ligação, como segue:

$$\nu = \frac{K1 * [L]}{1 + K1 * [L]} + \frac{K2 * [L]}{1 + K2 * [L]} \tag{7.6}$$

O trecho de código abaixo exemplifica o modelo no `R`, bem como suas principais linearizações diagnósticas.

```
# Heterogeneidade de sítios de ligação

L <- c(0.1, 0.2, 0.5, 1, 5, 10, 20) * 1e-6
Kd1 <- 2e-6
n1 <- 1
Kd2 <- 2e-8
n2 <- 1
v <- (n1 * L / (Kd1 + L)) + (n2 * L / (Kd2 + L))

par(mfrow = c(2, 3)) # estabelece área de plot pra 6 gráficos
plot(L, v, type = "o", main = "Direto")
plot(log(L), v, type = "o", main = "Langmuir")
plot(1 / L, 1 / v, type = "o", main = "Klotz")
plot(v, v / L, type = "o", main = "Scatchard")
plot(L, L / v, type = "o", main = "Woolf")
plot(log10(L), log10(v / (n1 + n2 - v)), type = "o", main = "Hill")
```

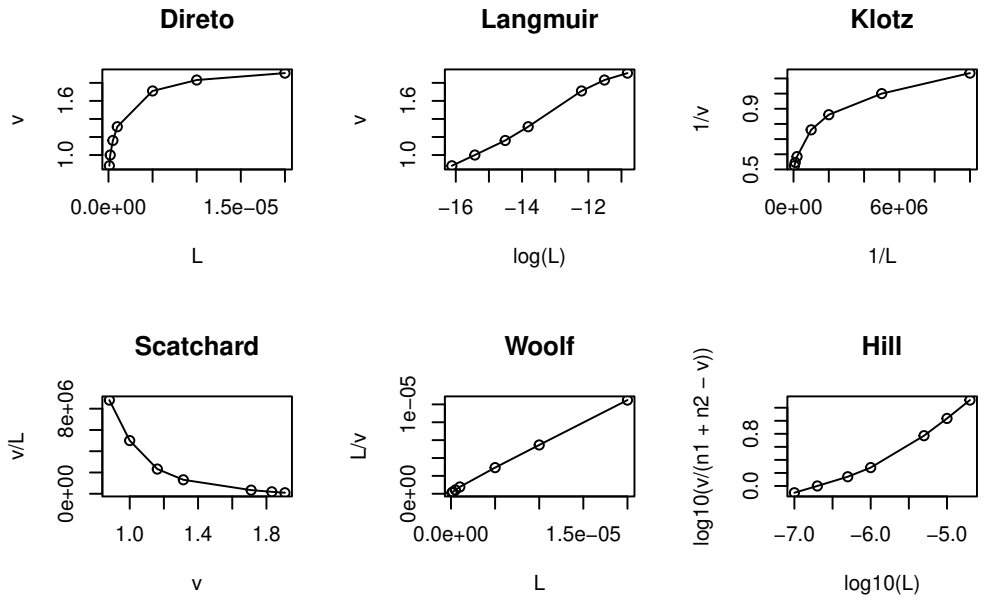


Figura 7.2: Modelo e linearizações para heterogeneidade de 2 conjuntos de sítios de ligação

```
# n1+n2=ntot no Hill
par(mfrow = c(1, 1)) # volta à janela gráfica normal
```

O *modelo de criação de novo sítio* - "one-site creator"; (Parsons e Vallner 1978) estabelece uma cooperatividade positiva resultante da produção de novos sítios para o ligante na proteína. Segue o modelo exemplificado e suas linearizações resultantes.

```
# Criação de novo sítio sob interação com ligante

L <- c(0.1, 0.2, 0.5, 1, 5, 10, 20) * 1e-6
Kd1 <- 2e-6
n1 <- 1
Kd2 <- 2e-5
n2 <- 1
nH <- 0.5
v <- (n1 * L * 1 / Kd1) / (1 + 1 / Kd1 * L) +
  ((n2 * 1 / Kd1 * 1 / Kd2 * L^2) / (1 + 1 / Kd1 * L) * (1 + 1 / Kd2 * L))
par(mfrow = c(2, 3)) # estabelece área de plot pra 6 gráficos
```

```
plot(L, v, type = "o", main = "Direto")
plot(log(L), v, type = "o", main = "Langmuir")
plot(1 / L, 1 / v, type = "o", main = "Klotz")
plot(v, v / L, type = "o", main = "Scatchard")
plot(L, L / v, type = "o", main = "Woolf")
plot(log10(L), log10(v / (n - v)), type = "o", main = "Hill")
```

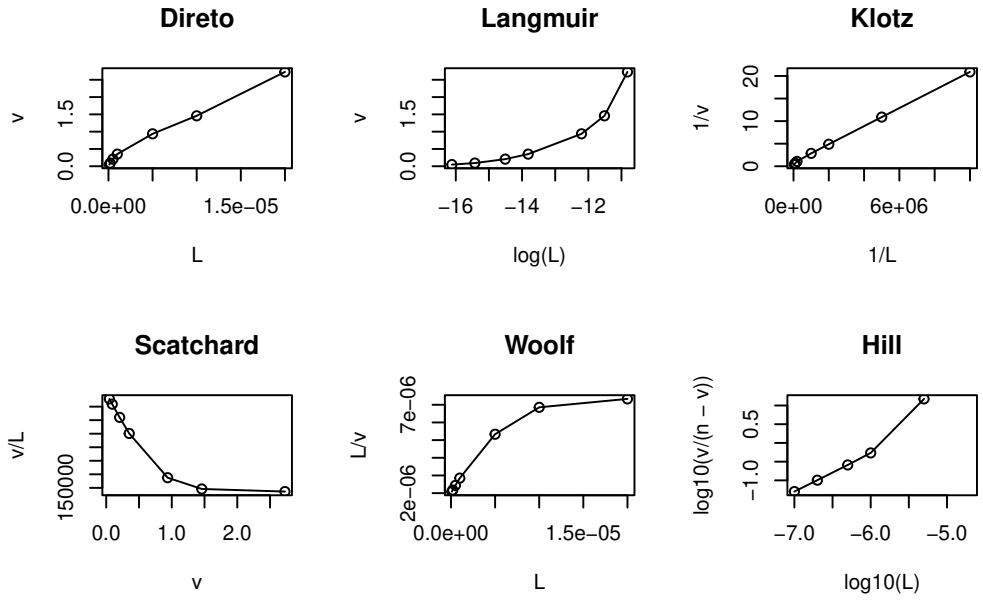


Figura 7.3: Modelo e linearizações para criação de novo sítio: *1-site creator*.

```
# n1+n2=ntot no Hill
par(mfrow = c(1, 1)) # volta à janela gráfica normal
```

Os *modelos de cooperatividade (negativa e positiva)* seguem um formalismo similar descrito para a ligação de oxigênio à hemoglobina em Capítulo 4 (Equação 4.2). Na *cooperatividade negativa* uma segunda molécula de ligante interage com a proteína com menor afinidade:

```
# Cooperatividade negativa em ligand-binding

L <- c(0.1, 0.2, 0.5, 1, 5, 10, 20) * 1e-6
Kd <- 2e-6
n <- 1
nH <- 0.5
v <- (n * L^nH / (Kd + L^nH))
par(mfrow = c(2, 3)) # estabelece área de plot pra 6 gráficos
plot(L, v, type = "o", main = "Direto")
plot(log(L), v, type = "o", main = "Langmuir")
plot(1 / L, 1 / v, type = "o", main = "Klotz")
plot(v, v / L, type = "o", main = "Scatchard")
plot(L, L / v, type = "o", main = "Woolf")
plot(log10(L), log10(v / (n - v)), type = "o", main = "Hill")
```

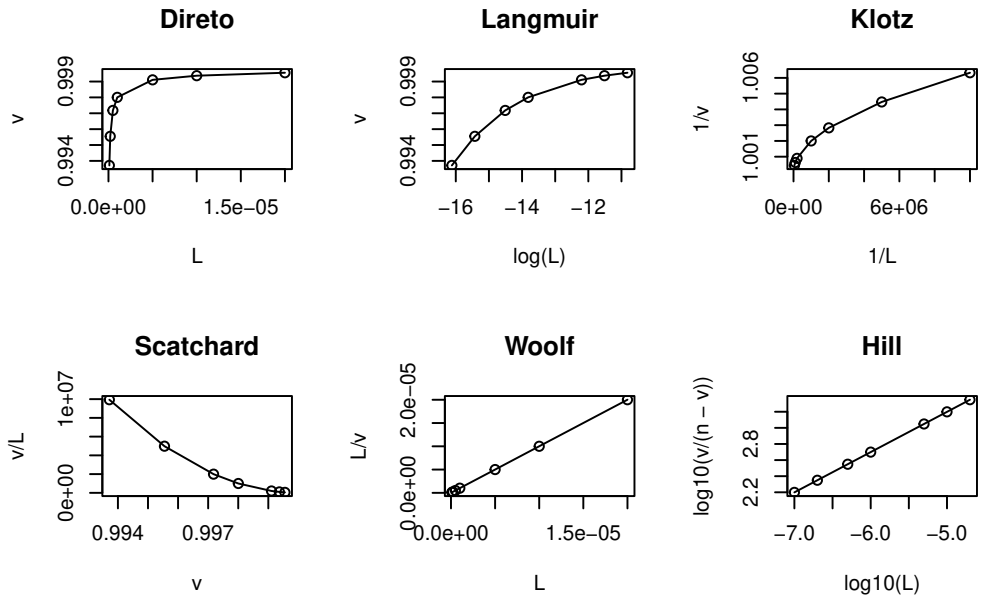


Figura 7.4: Modelo e linearizações para cooperatividade negativa de sítios de ligação.

```
# n1+n2=ntot no Hill
par(mfrow = c(1, 1)) # volta à janela gráfica normal
```

Já na *cooperatividade positiva*, uma segunda molécula de ligante interage com a proteína com maior afinidade que a primeira molécula (Parsons e Vallner 1978):

```
# Cooperatividade positiva em ligand binding

L <- c(0.1, 0.2, 0.5, 1, 5, 10, 20) * 1e-6
Kd <- 2e-6
n <- 1
nH <- 1.5
v <- (n * L^nH / (Kd + L^nH))
par(mfrow = c(2, 3)) # estabelece área de plot pra 6 gráficos
plot(L, v, type = "o", main = "Direto")
plot(log(L), v, type = "o", main = "Langmuir")
plot(1 / L, 1 / v, type = "o", main = "Klotz")
plot(v, v / L, type = "o", main = "Scatchard")
plot(L, L / v, type = "o", main = "Woolf")
plot(log10(L), log10(v / (n - v)), type = "o", main = "Hill")
```

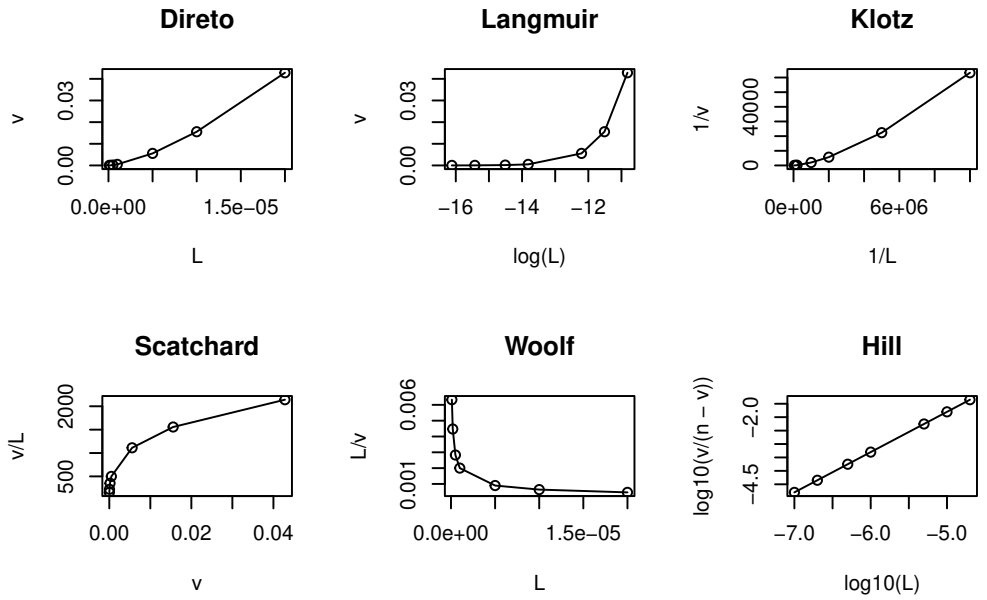


Figura 7.5: Modelo e linearizações para cooperatividade positiva de sítios de ligação.

```
# n1+n2=ntot no Hill
par(mfrow = c(1, 1)) # volta à janela gráfica normal
```

Observe que a inclinação do gráfico de *Hill* é inferior à unidade para a *cooperatividade negativa*, e superior a essa, para a *cooperatividade positiva*, e representa o mesmo parâmetro *nH* visto na Equação 4.2.

Ainda que sujeito à crítica por sua inconsistência estatística (variável dependente em ambos os eixos), a representação de Scatchard tem sido privilegiada ao longo de décadas como diagnóstico de modelos de interação ligante-proteína. Entre suas vantagens, aloca-se a possibilidade de facilmente distinguir-se o modelo de cooperatividade positiva (aclive) do de heterogeneidade de sítios de ligação (declive abrupto) ou de cooperatividade negativa (declive suave).

Sec 7.2 Ajuste Não-Linear Em Interação Ligante-Proteína

Ajustes diretos da equação não linear dos modelos de interação também podem ser efetuados como fora realizado para a equação de Michaelis-Menten no Capítulo 5. Exemplificando, pode-se simular a obtenção de dados experimentais de *binding* pelo trecho a seguir, utilizando-se o comando 'runif' (*random uniform*) para geração de sequência aleatória (como fora realizado no Capítulo 5). Para ilustrar, segue a Figura 7.6.

```
# Isoterma de Interação Ligante-Proteína
n <- 1
Kd <- 10
L <- 120
i <- 3
L <- seq(0, L, i)
v <- (L * n) / (Kd + L) + rnorm(40, 0, 0.1)
plot(L, v)
```

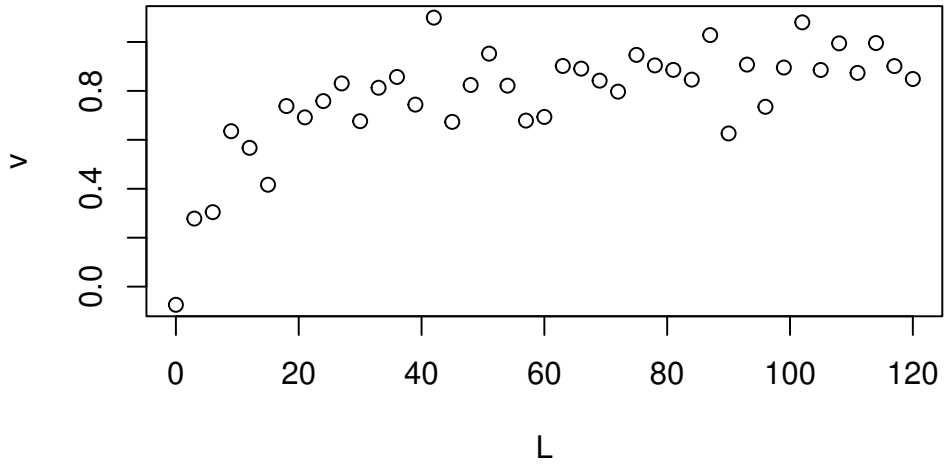


Figura 7.6

Observe que o comando 'rnorm' adiciona um erro de distribuição normal aos dados. Outra forma para simulação desses é dada abaixo, introduzindo-se o comando 'runif' de geração de números aleatórios.

```
# Simulação de dados de interação bimolecular (1 sítio)

# Simulação de dados
set.seed(20160227) # estabelece semente para geração de números aleatórios
L <- seq(0, 50, 1)
PL <- ((runif(1, 10, 20) * L) / (runif(1, 0, 10) + L)) + rnorm(51, 0, 1)
# 1. runif(n,min,max); quando sem atributos, considera-se min=0  e max=1
# 2. rnorm(no. pontos,media,desvio) - erro aleatório de distribuição normal
plot(L, PL, xlab = "L", ylab = "PL")
```

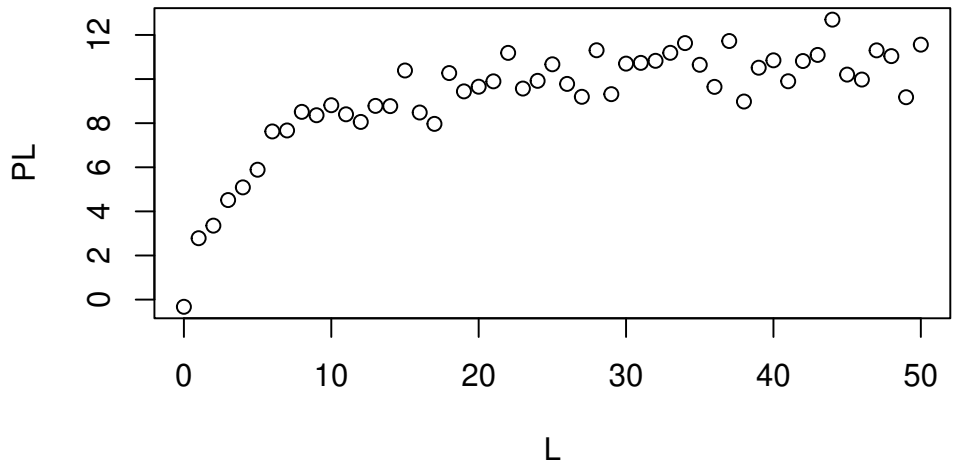


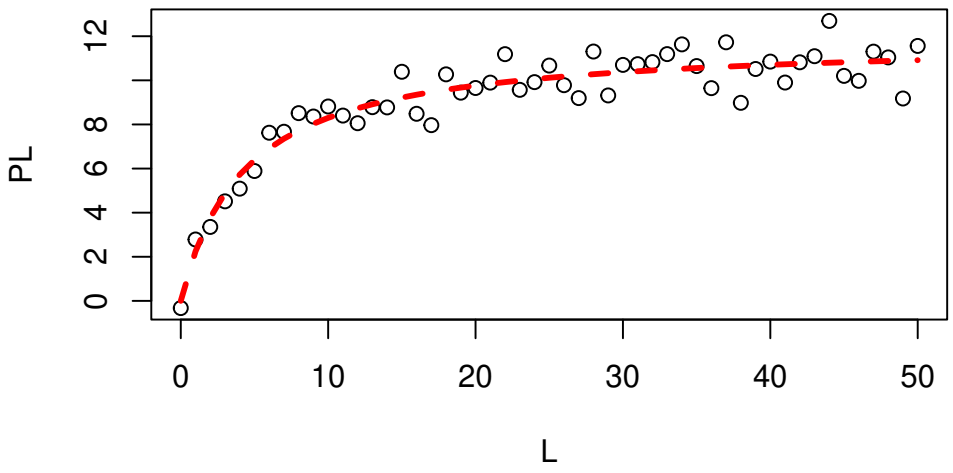
Figura 7.7: Dados simulados para isoterma de interação bimolecular.

Agora precisamos utilizar o comando 'nls' para o ajuste não linear, sobreposição da curva teórica, e tabela estatística de resultados:

```
# Ajuste não linear em ligand binding
m <- nls(PL ~ n * L / (Kd + L), start = list(n = 1, Kd = 1))

# Coef. de correlação
cor(PL, predict(m)) # Coeficiente de correlação de Pearson
```

```
[1] 0.9496598
```

```
# Gráfico de dados e simulação
plot(L, PL)
lines(L, predict(m), lty = 2, col = "red", lwd = 3)
```



```
summary(m)
```

```
Formula: PL ~ n * L/(Kd + L)

Parameters:
   Estimate Std. Error t value Pr(>|t|)
n   11.8478     0.2620  45.216  < 2e-16 ***
Kd   4.2778     0.5113   8.366  5.3e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7826 on 49 degrees of freedom

Number of iterations to convergence: 6
Achieved convergence tolerance: 1.554e-06
```

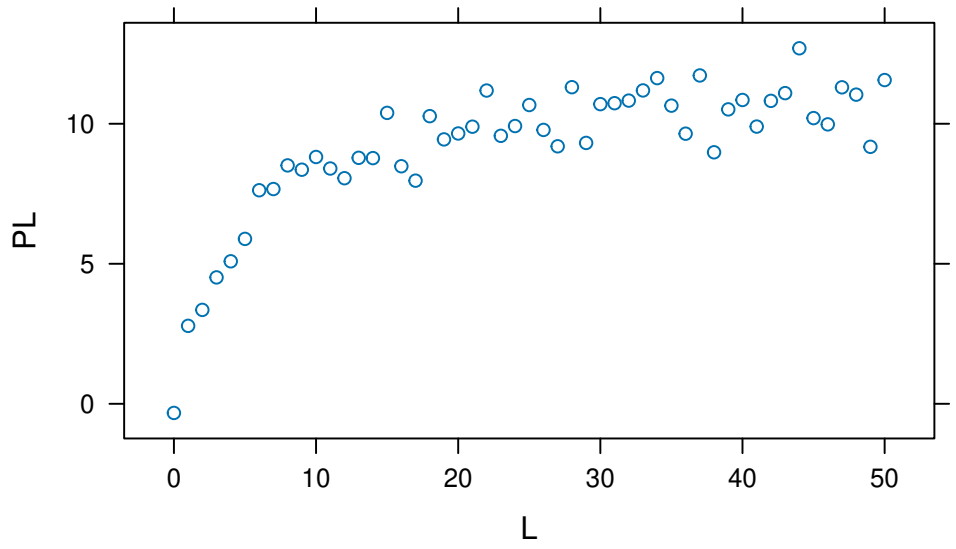
Sec 7.3 Sistemas Gráficos no R

A última curva de simulação obtida o foi junto à biblioteca padrão para manipulação de gráficos da instalação do 'R', **Graphics**. Trata-se um conjunto de funções amplo também utilizado por vários outros pacotes do ambiente. Contudo, existem no 'R' diversas outras bibliotecas para elaboração de gráficos, dentre os quais vale destacar o *Lattice*, também incluido na instalação padrão, e o *ggplot2*. Ambos os sistemas geram resultados com melhor estética e flexibilidade gráfica que a biblioteca *Graphics* padrão, e possuem empregos e semânticas distintas entre si.

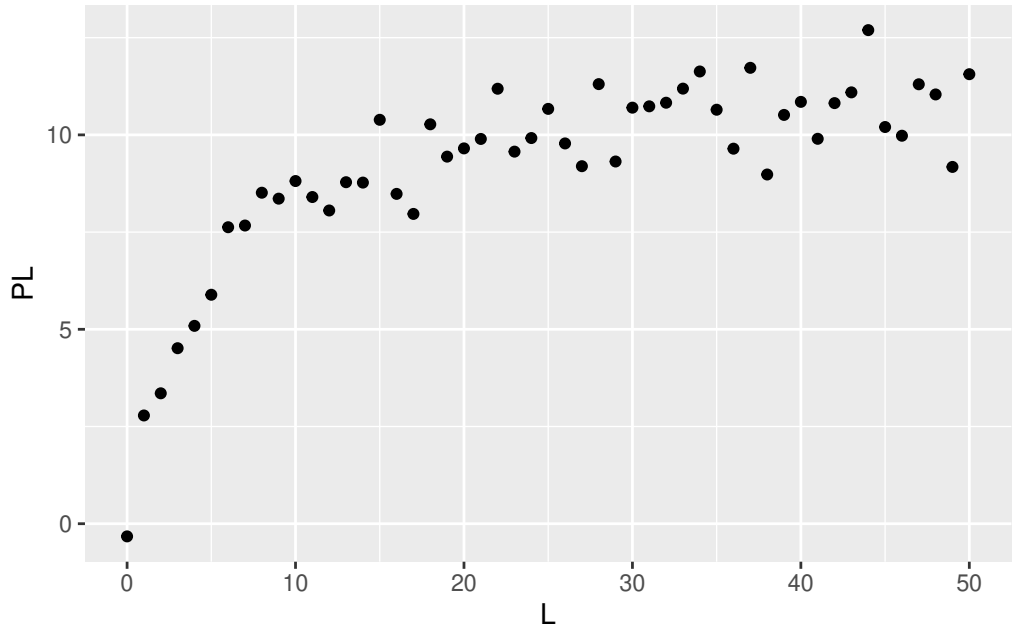
O sistema *Lattice* (Sarkar 2008) é baseado no sistema *Trellis* para representação gráfica de dados multivariados. Sua força está na representação de dados em paineis contendo subgrupos e, embora tenha sintaxe menos intuitiva e por vezes mais elaborada que o pacote *Graphics*, produz um grafismo superior a esse com poucos cliques de teclado. De modo geral, o *Lattice* produz o gráfico dentro do próprio algoritmo, de modo diferente aos sistemas *Graphics* (pode-se acumular linhas sucessivas de modificação do gráfico) ou *ggplot2*.

Por outro lado a biblioteca *ggplot2* é baseada na gramática de gráficos (Wickham 2011), e produz o gráfico utilizando uma única linha de comando que combina camadas sobrepostas, de modo similar à aplicativos de manipulação de imagens (ex: *Inkscape*, *Gimp*, *Corel Draw*, *Photoshop*). Dessa forma é possível alterar cada ítem do gráfico em suas camadas específicas (tema, coordenadas, facets, estatísticas, geometria, estética, dados). Exemplificando o resultado gráfico da curva de simulação acima de *binding* para *Lattice* e *ggplot2*:

```
# Os sistemas lattice e ggplot2

# Simulação de dados

set.seed(20160227) # estabelece semente para geração de números aleatórios
L <- seq(0, 50, 1)
PL <- ((runif(1, 10, 20) * L) / (runif(1, 0, 10) + L)) + rnorm(51, 0, 1)
# 1. runif(n,min,max); quando sem atributos, considera-se min=0  e max=1
# 2. rnorm(no. pontos,media,desvio) - erro aleatório de distribuição normal

# Produção do gráfico com sistema Lattice
library(lattice)
xyplot(PL ~ L)
```




```
# Produção do gráfico com sistema ggplot2
library(ggplot2)
qplot(L, PL)
```



Como a percepção de peculiaridades dos sistemas gráficos se revela melhor com dados mais elaborados, as ilustrações a seguir utilizarão o conjunto de dados 'Puromycin', que integra a biblioteca 'datasets' do 'R'. Os dados apresentam a velocidade de reação enzimática sobre um substrato em células tratadas e não tratadas com puromicina.

O código abaixo retorna a plotagem, ajuste não linear e resultados obtidos com a biblioteca *Graphics* padrão, e foi extraído do própria documentação do *dataset*.

```
library(datasets)

# O sistema base `graphics`

plot(rate ~ conc,
  data = Puromycin, las = 1,
```

```
  xlab = "[S], mM",
  ylab = "v (contagem/min/min)",
  pch = as.integer(Puromycin$state),
  col = as.integer(Puromycin$state),
  main = "Ilustração de Ajuste Com Graphics"
)

## Ajuste da equação de Michaelis-Mentem
fm1 <- nls(rate ~ Vm * conc / (K + conc),
  data = Puromycin,
  subset = state == "treated",
  start = c(Vm = 200, K = 0.05)
)
fm2 <- nls(rate ~ Vm * conc / (K + conc),
  data = Puromycin,
  subset = state == "untreated",
  start = c(Vm = 160, K = 0.05)
)
summary(fm1)
```

```
Formula: rate ~ Vm * conc/(K + conc)

Parameters:
    Estimate Std. Error t value Pr(>|t|)
Vm 2.127e+02  6.947e+00  30.615 3.24e-11 ***
K  6.412e-02  8.281e-03   7.743 1.57e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.93 on 10 degrees of freedom

Number of iterations to convergence: 5
Achieved convergence tolerance: 8.768e-06
```

```
summary(fm2)
```

```
Formula: rate ~ Vm * conc/(K + conc)

Parameters:
    Estimate Std. Error t value Pr(>|t|)
Vm 1.603e+02  6.480e+00  24.734 1.38e-09 ***
K  4.771e-02  7.782e-03   6.131 0.000173 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.773 on 9 degrees of freedom

Number of iterations to convergence: 5
Achieved convergence tolerance: 4.473e-06
```

```
## Adição de linhas de ajuste ao plot
conc <- seq(0, 1.2, length.out = 101)
lines(conc, predict(fm1, list(conc = conc)), lty = 1, col = 1)
lines(conc, predict(fm2, list(conc = conc)), lty = 2, col = 2)
legend(0.8, 120, levels(Puromycin$state),
  col = 1:2, lty = 1:2, pch = 1:2
)
```

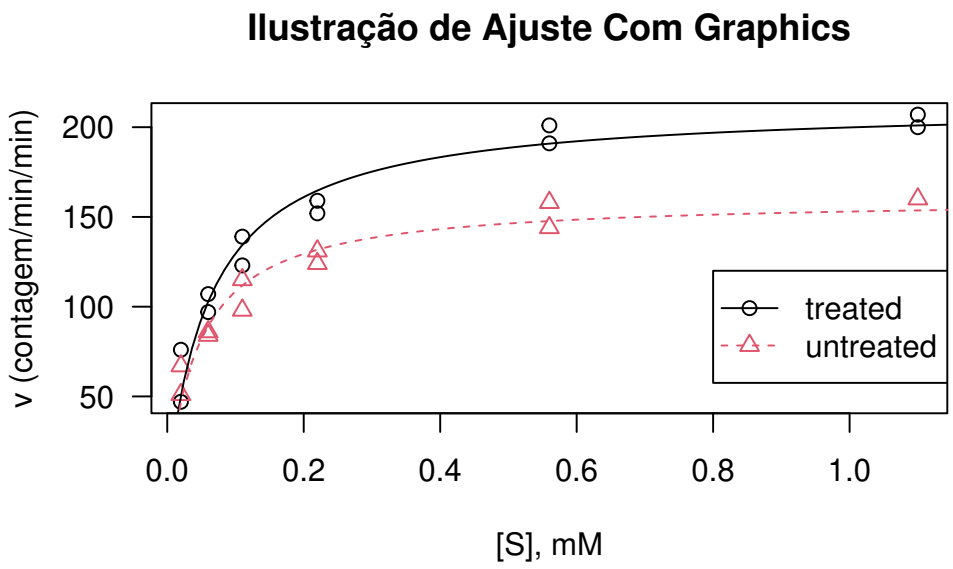


Figura 7.8: Plotagem e análise com `graphics`.

O sistema *ggplot2*, por sua vez, exige que os comandos sejam elencados em camadas justapostas intercaladas com o sinal "+", como segue:

```
# Gráfico e análise com ggplot2
library(datasets)
p <- ggplot(data = Puromycin, aes(conc, rate, color = state)) +
  geom_point() +
  geom_smooth(
    method       = "nls",
    formula      = y ~ Vm * x / (Km + x),
    method.args  = list(start = list(Vm = 200, Km = 0.1)),
    se           = FALSE
  ) # expressão que define o plot
p # variável que apresenta o plot
```

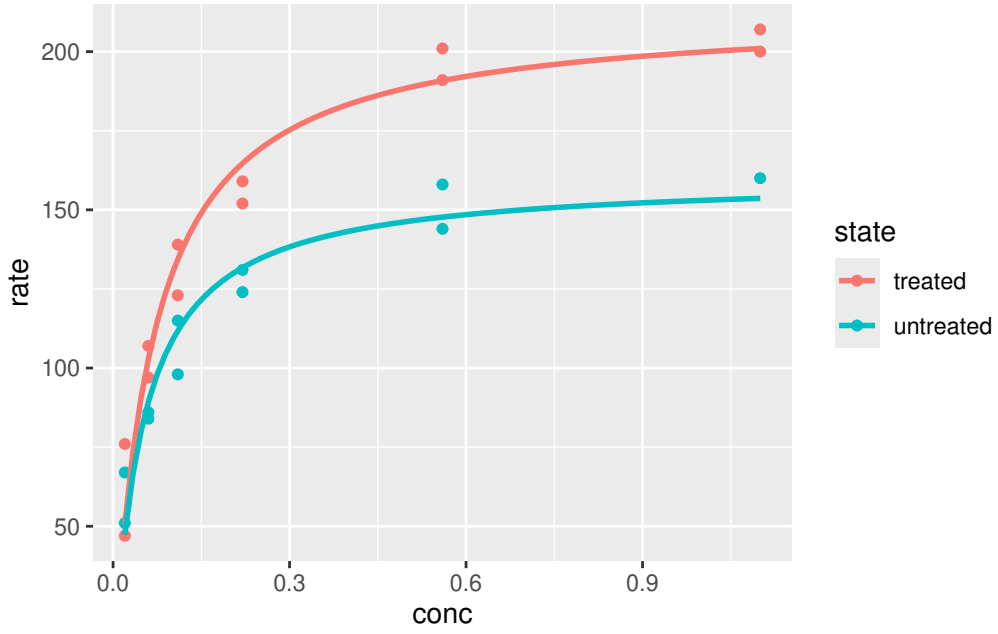


Figura 7.9: Plotagem e análise com ggplot2.

Perceba o menor número de instruções do *script* para a produção do gráfico. Além disso, e diferente do *Graphics*, *ggplot2* permite adicionar camadas à linha de comandos principais, e apresentar os dados multivariados em paineis (funções 'facet_grid' e 'facet_wrap'), sem a necessidade de se utilizar o comando 'mfrow' ou 'mfcol' visto no capítulo, como segue:

```
p + facet_grid(rows = vars(state))
```

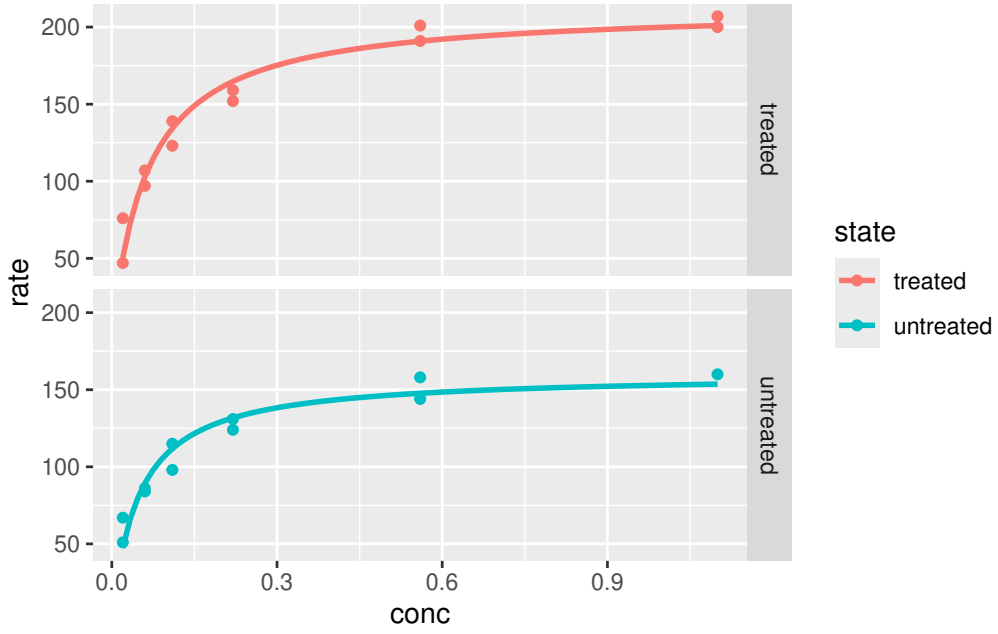


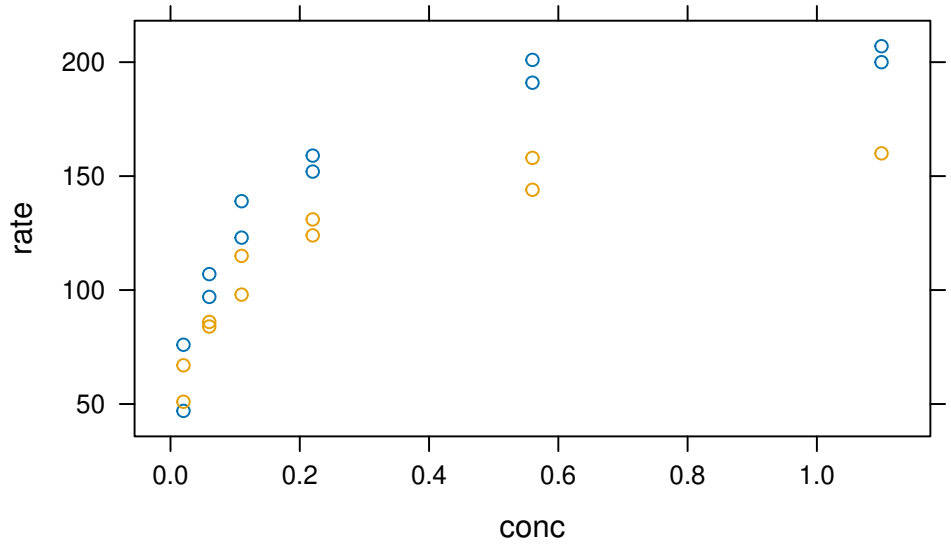
Figura 7.10: Plotagem e análise com ggplot2 - paineis (faceting).

Lattice também possui uma economia de instruções em relação à *Graphics*. Não considerando o ajuste estatístico, os grupos podem ser apresentados simplesmente utilizando-se a fórmula:

7

$$xyplot(y \sim x \mid groups = z) \tag{7.7}$$

```
library(lattice)
xyplot(rate ~ conc, data = Puromycin, groups = state)
```



E para a representação dos ajustes não lineares:

```
# Um gráfico com dataset para lattice

library(nlme)
n1 <- nlsList(rate ~ Vmax * conc / (Km + conc) | state,
              data = Puromycin, start = list(Vmax = 200, Km = 0.1))
summary(n1)
xyplot(rate ~ conc,
  groups = state, data =
    Puromycin
) +
  layer(panel.curve(Vmax[1] * x / (Km[1] + x), col = 1),
    data = as.list(coef(n1))
  ) +
  layer(panel.curve(Vmax[2] * x / (Km[2] + x), col = 2),
    data = as.list(coef(n1))
  )
```

A biblioteca *Lattice* também permite a apresentação em paineis; diferente de *ggplot2*, contudo, o gráfico é gerado algoritmicamente, sem a sobreposição de comandos:

```
# Gráfico e análise não linear com lattice

library(nlme) # pacote quer permite regressão não linear com subgrupos
nonlinLatt <- nlsList(rate ~ Vmax * conc /
                        (Km + conc) | state,
                      start = list(Vmax = 200, Km = 0.1), data = Puromycin)
summary(nonlinLatt)
```

```
Call:
  Model: rate ~ Vmax * conc/(Km + conc) | state
   Data: Puromycin
```

```
Coefficients:
   Vmax
          Estimate Std. Error  t value     Pr(>|t|)
treated   212.6836   6.608088 32.18535 3.241151e-11
untreated 160.2800   6.896011 23.24242 1.384612e-09
   Km
            Estimate  Std. Error  t value     Pr(>|t|)
treated   0.06412110 0.007876774 8.140529 0.0000156514
untreated 0.04770812 0.008281147 5.761052 0.0001727056

Residual standard error: 10.40003 on 19 degrees of freedom
```

```
xyplot(rate + fitted(nonlinLatt) ~ conc | state,
  data = Puromycin,
  type = c("p", "l"), distribute.type = TRUE, col.line = "red",
  ylab = "rate"
)
```

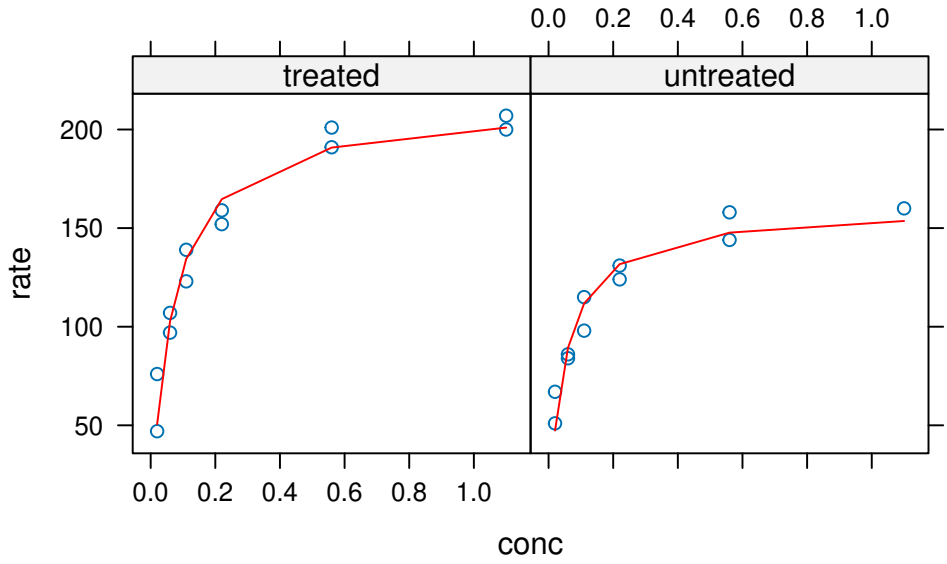


Figura 7.11: Plotagem e análise com Lattice - ajuste externo e paineis.

Das muitas diferenças que os pacotes *ggplot2* e *Lattice* apresentam, há uma que vale a pena ressaltar. Como visto acima, o *ggplot2* realiza o ajuste não linear dentro da linha de comandos de geração do gráfico, ao passo que o *Lattice* permite fazê-lo fora da linha. Isso é inerente do *ggplot2*, uma biblioteca desenhada para a produção de gráficos, e não para análises computacionais. Dessa forma o algoritmo que permite o ajuste não linear pelo *ggplot2*, ainda que seja o mesmo 'nls' já trabalhado, não expressa seus resultados explicitamente (embora haja formas de "pescá-los" utilizando-se outros pacotes).

De certa forma, ainda que o *Lattice* exija uma curva de aprendizado menos intutiva, ele permite que se utilize os resultados estatísticos obtidos anteriormente para inclusão no algoritmo de plotagem. Isso é vantajoso quando se deseja outros algoritmos estatísticos para ajuste, como acima, ou mesmo sua flexibilização, além do 'nls' incluido em *ggplot2*. Não obstante, o *Lattice* também permite que se inclua a linha de ajuste dentro do próprio algoritmo, como abaixo:

```
# Ajuste não linear em paineis (Lattice)

xyplot(rate ~ conc | state,
  data = Puromycin,
  panel = function(x, y, ...) {
    panel.xyplot(x, y, ...)
```

```
    n3 <- nls(y ~ Vmax * x / (Km + x), data = Puromycin,
              start = list(Vmax = 200, Km = 0.1))
    panel.lines(seq(0.02, 1.1, 0.02),
                predict(n3, newdata = data.frame(x = seq(0.02, 1.1, 0.02))),
                col.line = 2)
  },
  xlab = "conc", ylab = "rate"
)
```

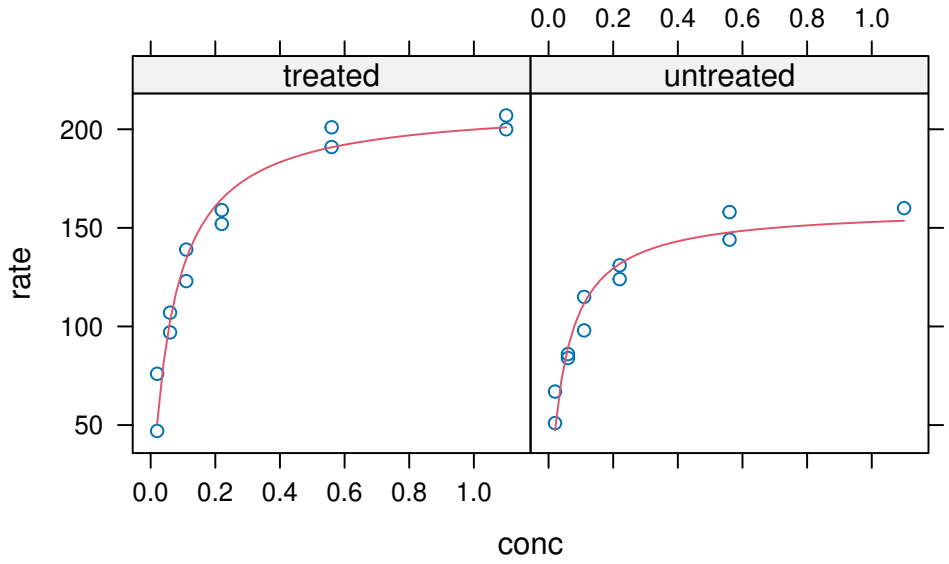


Figura 7.12: Plotagem e análise com Lattice - ajuste interno e paineis.

Sec 7.4 Solução Numérica Para o Equilíbrio de Complexos Ligante-Proteína

Como visto no capítulo Capítulo 3, por vezes uma *solução numérica* pode ser empregada quando a *solução analítica* não converge, ou quando estamos diante de uma equação mais complexa. Nesse sentido a formação de complexos de que trata este subtítulo pode também ser tratada por uma solução numérica.

Usualmente o tratamento dado para a solução numérica envolve encontrar as raízes de uma equação ou sistema de equações, ou seja:

$$F(x) = 0 \tag{7.8}$$

$$P + L \rightleftarrows PLKd = \frac{[P][L]}{[PL]} Então,\ [PL] = \frac{[P][L]}{Kd} \tag{7.9}$$

$$\begin{cases} [Pt] = [P] + [PL] = [P] + \frac{[P][L]}{Kd} \\ [Lt] = [L] + [PL] = [L] + \frac{[P][L]}{Kd} \end{cases} \tag{7.10}$$

Assim, tem-se um sistema de equações lineares nos parâmetros (*[P]* livre, *[L]* livre, e complexo *[PL]*) que pode ser solucionado pelo `R` por diversas maneiras, uma das quais pela função de minimização `rootSolve`:

```
# Cálculo de L, P, e PL em interação biomolecular para 1 conjunto de sítios
# de mesma afinidade
library(rootSolve)
Pt <- 1
```

```
Lt <- 10
Kd <- 4

# Modelo
model <- function(x) c(x[1] + (x[1] * x[2]) / Kd - Pt, x[2] +
                         (x[1] * x[2]) / Kd - Lt, Pt - x[1] - x[3])
# o modelo acima deve conter uma lista de equações cuja igualdade é zero,
# ou seja, f(x)=0
(ss <- multiroot(model, c(1, 1, 1))) # comando de execução do
```

```
$root
[1] 0.3007353 9.3007353 0.6992647

$f.root
[1] 5.466137e-08 5.463051e-08 5.191070e-12

$iter
[1] 5

$estim.precis
[1] 3.643236e-08
```

```
# rootSolve (sementes pro algoritmo)
```

Dessa forma os valores resultantes (*f.root*) quando *[Lt]* = 10 são apresentados como *[P]* = 0,3, *[L]* = 9,3 e *[PL]* = 0,7.

Outras soluções numéricas permitem um maior controle sobre o algoritmo empregado, tais como a função `optim` do `R` (limites de busca da solução, emprego de vetores, por ex). Para isso será exemplificado a mesma situação acima, embora apresentando uma variação do formalismo que relaciona P, L e PL :

$$\begin{cases} [PL] = [Pt] - [L] \\ [PL] = [Lt] - [L] \\ [P] * [L] = Kd * [PL] \end{cases} \tag{7.11}$$

Dessa forma pode-se contruir uma relação quadrática envolvendo as três incógnitas:

$$([Pt] - [L] - [PL])^2 + ([Lt] - [L] - [PL])^2 + ([P] * [L] - Kd * [PL])^2 \tag{7.12}$$

Aplicando-se o algoritmo de minimização `optim` do `R`:

```
# Cálculo de L, P, PL em interação para 1 sítio

model2 <- function(x, Pt, Lt, K) {
  L <- x[1]
  P <- x[2]
  PL <- x[3]
  (Pt - P - PL)^2 + (Lt - L - PL)^2 + (P * L - Kd * PL)^2
} # declaração da função
Pt <- 1
Lt <- 10
Kd <- 4 # parâmetros da função
sol2num <- optim(c(0.5, 1, .5), model2, method = "L-BFGS-B",
                 lower = c(0, 0, 0), upper = c(Lt, Pt, Pt), Pt = Pt, Lt = Lt)
# método BFGS permite bounds (lower, upper)
sol2num$par # LF, PF, PL calculados
```

```
[1] 9.3007349 0.3007355 0.6992652
```

Perceba que são os mesmos resultados anteriores, embora com maior controle da solução. Agora pode-se utilizar essa minimização para criar um vetor de soluções para as três quantidades, como segue:

```
# Declaração da função
bind1 <- function(x, Pt, Lt, Kd) {
  L <- x[1]
  P <- x[2]
  PL <- x[3]
  (Pt - P - PL)^2 + (Lt - L - PL)^2 + (P * L - Kd * PL)^2
}

# Parâmetros da função
Pt <- 1
Lt <- c(5, 10, 20)
Kd <- 4

# Minimização (parâmetros para que a função acima dê zero)
y <- function(i) {
  optim(c(1, 1, 1), bind1,
    method =
      "L-BFGS-B", lower = c(0, 0, 0), upper = c(
      Lt[i], Pt,
      Pt
    ), Lt = Lt[i], Pt = Pt, Kd = Kd
  )
}

# Resultados em matriz
ypar <- function(i) y(i)$par
yp <- matrix(
  nrow = length(Lt), ncol = 2 + length(Kd),
  byrow = T
)
for (i in 1:length(Lt)) yp[i, ] <- y(i)$par
colnames(yp) <- c("L", "P", "PL")
rownames(yp) <- c("5", "10", "20")
yp
```

```
           L         P         PL
5   4.472136 0.4721359 0.5278640
10  9.300736 0.3007344 0.6992634
20 19.172624 0.1726180 0.8273844
```

Sec 7.5 Cinética de Interação Ligante-Proteína e Solução Numérica

Sob o mesmo princípio da solução numérica apresentada no ítem anterior para o equilíbrio da interação ligante-proteína, o `R` permite solução de mesma natureza para a cinética da formação dos complexos, ou seja, os teores de P, L e PL observados no tempo. Nesse caso pode-se desenvolver outras relações a partir da Equação 7.1. Tomando-se por base que no equilíbrio as taxas cinéticas de k_{on} e k_{off} se igualam (*steady-state*), pode-se relacionar algumas equações diferenciais para a *associação*, bem como para a *dissociação* dos complexos:

$$Para\ a\ associação: \begin{cases} \frac{d[PL]}{dt} = k_{on} * [L] * [P] \\ \frac{d[L]}{dt} = -k_{on} * [L] * [P] \\ \frac{d[P]}{dt} = -k_{on} * [L] * [P] \end{cases} \tag{7.13}$$

$$Para\ a\ dissociação: \begin{cases} \frac{d[PL]}{dt} = -k_{off} * [PL] \\ \frac{d[L]}{dt} = k_{off} * [PL] \\ \frac{d[P]}{dt} = k_{off} * [PL] \end{cases} \tag{7.14}$$

Assim, as taxas globais resultantes para cada quantidade (taxa líquida) envolverá a soma das taxas de associação e dissociação de cada:

$$Taxas\,líquidas: \begin{cases} \frac{d[PL]}{dt} = k_{on} * [L] * [P] - k_{off} * [PL] \\ \frac{d[L]}{dt} = -k_{on} * [L] * [P] + k_{off} * [PL] \\ \frac{d[P]}{dt} = -k_{on} * [L] * [P] + k_{off} * [PL] \end{cases} \tag{7.15}$$

A solução para esse sistema final de equações diferenciais (taxas líquidas) envolve resolver a variação de cada quantidade (Δx) num determinado intervalo de tempo (Δt), tal que:

$$\Delta x = f(x) * \Delta t \tag{7.16}$$

Para isso é necessário utilizar uma biblioteca do `R` que permita a solução de um sistema de equações diferenciais. Entre as muitas soluções (`odeintr`, `pracma`, `rODE`), o emprego da biblioteca `deSolve`, que utiliza uma função para integração do sistema por algoritmo de *Runge-Kutta de 4a. ordem*:

```
# Cinética de interação ligante-protéina para 1 conjunto de sítios
library(deSolve)

# Condições experimentais
tempo <- seq(0, 100) # intervalo de tempo
parms <- c(kon = 0.02, koff = 0.001) # parâmetros do estado estacionário
# da interação (uM^-1*s^-1 e s^-1, respectivamente)
val.inic <- c(L = 0.8, P = 1, PL = 0) # valores iniciais, uM

# Integração do sistema por Runge-Kutta de 4a. ordem
solNumKin <- function(t, x, parms) {
  # definição da lista de parâmetros
  L <- x[1] # ligante
  P <- x[2] # proteína
  PL <- x[3] # complexo

  with(as.list(parms), {
    # definição da lista de equações diferenciais
    dL <- -kon * L * P + koff * PL
    dP <- -kon * L * P + koff * PL
    dPL <- kon * L * P - koff * PL
    res <- c(dL, dP, dPL)
    list(res)
  })
}

sol.rk4 <- as.data.frame(rk4(
  val.inic, tempo, solNumKin,
  parms
)) # rotina para Runge-Kutta 4a. ordem

# Gráfico
plot(sol.rk4$time, sol.rk4$L, type = "l", xlab = "tempo",
     ylab = "[composto], uM")
legend("topright", c("L", "P", "PL"),
       text.col = c("black", "red", "blue"), bty = "n", lty = c(1, 2, 3))
lines(sol.rk4$time, sol.rk4$P, type = "l", lty = 2, col = 2, lwd = 1.5)
lines(sol.rk4$time, sol.rk4$PL, type = "l", lty = 3, col = 3, lwd = 1.5)
```

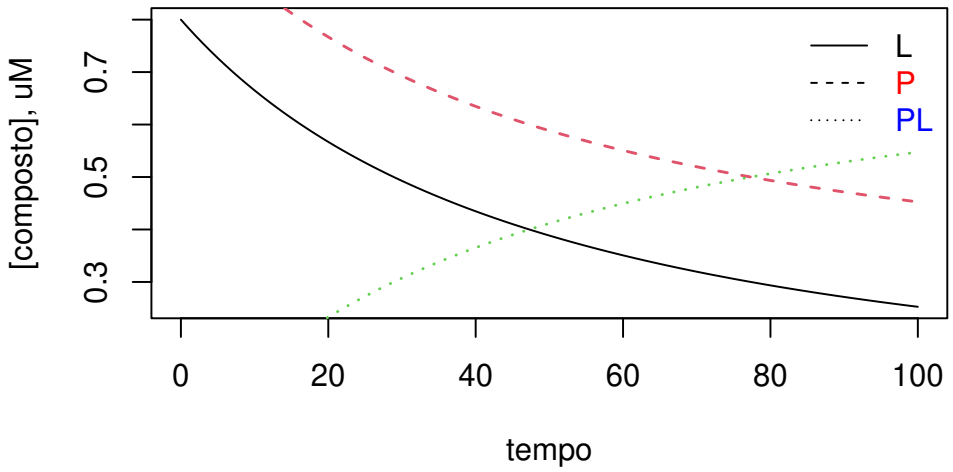


Figura 7.13: Teores de ligante e proteína livres (L e P), bem como do complexo PL apresentados ao longo do tempo de acordo com o método Runge-Kutta de 4a. ordem para solução de equações diferenciais.

Sec 8.1 Análise de sequências

Ácidos nucleicos podem ser considerados como sequências alfabéticas de 1 letra (bases), 2 letras (dinucleotídeo), ou de 3 letras (codon). Tomando-se o exemplo da lisozima de galinha:

1. Acessa-se o banco de dados do *NCBI - National Center for Biotechnology Information* [1];
2. Seleciona-se o banco de dados *Nucleotide*;
3. Digita-se a sequência de interesse; ex: "hen egg" lysozyme";
4. Seleciona-se LYZF1 (ou a referência de sequência do NCBI: NM_205281.2); Obs: Com o número de referência é possível acessar o conteúdo desejado a partir de uma consulta simples no Google.
5. Procura-se pela sequência referenciada em FASTA
6. O sítio apresentará a sequência nucleotídica para a lisozima, que pode ser copiada/colada no `R`, ou exportada como arquivo em "Send to. . . .File".

Agora precisa-se converter esta sequência de letras (*string*) em um vetor de bases que possa ser lido pelo `R`, e omitindo-se a quebra de linha. Isso pode ser agilizado com o pacote `seqinr` ou `TmCalculator` pela função `s2c` (converte *string* em vetor de *strings*; `c2s` faz o oposto). Ou também pelo pacote `stringr`:

```
# Conversão de sequência alfabética em vetor de bases

library(stringr)
liso.nucl <- "GCAGTCCCGCTGTGTGTACGACACTGGCAACATGAGGTCTTTGCTAATCTTGGTGC
TTTGCTTCCTGCCCCTGGCTGCTCTGGGGAAAGTCTTTGGACGATGTGAGCTGGCAGCGGCTATGAAGCG
TCACGGACTTGATAACTATCGGGGATACAGCCTGGGAAACTGGGTGTGTGCCGCAAAATTCGAGAGTAAC
TTCAACACCCAGGCTACAAACCGTAACACCGATGGGAGTACCGACTACGGAATCCTACAGATCAACAGCC
GCTGGTGGTGCAACGATGGCAGGACCCCAGGCTCCAGGAACCTGTGCAACATCCCGTGCTCAGCCCTGCT
GAGCTCAGACATAACAGCGAGCGTGAACTGCGCGAAGAAGATCGTCAGCGATGGAAACGGCATGAACGCG
TGGGTCGCCTGGCGCAACCGCTGCAAGGGCACCGACGTCCAGGCGTGGATCAGAGGCTGCCGGCTGTGAG
GAGCTGCCGCGCCCGGCCCGCCCGCTGCACAGCCGGCCGCTTTGCGAGCGCGACGCTACCCGCTTGGCAG
TTTTAAACGCATCCCTCATTAAAACGACTATACGCAAACGCC"

liso.nucl <- unlist(strsplit(liso.nucl, ""))
# converte sequência gênica de uma palavra em nucleotídios separados
liso.nucl[1:100] # uma amostra do resultado
```

```
  [1] "G"  "C"  "A"  "G"  "T"  "C"  "C"  "C"  "G"  "C"  "T"  "G"  "T"  "G"  "T"
 [16] "G"  "T"  "A"  "C"  "G"  "A"  "C"  "A"  "C"  "T"  "G"  "G"  "C"  "A"  "A"
 [31] "C"  "A"  "T"  "G"  "A"  "G"  "G"  "T"  "C"  "T"  "T"  "T"  "G"  "C"  "T"
 [46] "A"  "A"  "T"  "C"  "T"  "T"  "G"  "G"  "T"  "G"  "C"  "\n" "T"  "T"  "T"
 [61] "G"  "C"  "T"  "T"  "C"  "C"  "T"  "G"  "C"  "C"  "C"  "C"  "T"  "G"  "G"
 [76] "C"  "T"  "G"  "C"  "T"  "C"  "T"  "G"  "G"  "G"  "G"  "A"  "A"  "A"  "G"
 [91] "T"  "C"  "T"  "T"  "T"  "G"  "G"  "A"  "C"  "G"
```

```
liso.nucl <- liso.nucl[liso.nucl != "\n"]
# elimina a quebra de linha do resultado anterior
liso.nucl[1:100] # uma amostra do resultado sem os "\n"
```

[1] NCBI. https://www.ncbi.nlm.nih.gov/protein

```
 [1] "G" "C" "A" "G" "T" "C" "C" "C" "G" "C" "T" "G" "T" "G" "T" "G" "T" "A"
[19] "C" "G" "A" "C" "A" "C" "T" "G" "G" "C" "A" "A" "C" "A" "T" "G" "A" "G"
[37] "G" "T" "C" "T" "T" "T" "G" "C" "T" "A" "A" "T" "C" "T" "T" "G" "G" "T"
[55] "G" "C" "T" "T" "T" "G" "C" "T" "T" "C" "C" "T" "G" "C" "C" "C" "C" "T"
[73] "G" "G" "C" "T" "G" "C" "T" "C" "T" "G" "G" "G" "G" "A" "A" "A" "G" "T"
[91] "C" "T" "T" "T" "G" "G" "A" "C" "G" "A"
```

Com a sequência gênica em mãos pode-se avaliar um extenso conjunto de propriedades ou manipular o vetor de bases, tal como referenciado em alguns pacotes do `R` (`seqinr`, `DNASeqtest`, `haplotypes`, `rDNAse`). Também pode-se proceder algum manuseio mais simples para o gene selecionado, como abaixo:

```
# Alguns cálculos manuais com a sequência de bases

length(liso.nucl[liso.nucl == "A"])
```

```
[1] 133
```

```
# quantifica as bases de purina na sequência

table(liso.nucl) # contagem de cada nucleotídio
```

```
liso.nucl
  A   C   G   T
133 173 174 108
```

```
library(seqinr)
liso.nucl2 <- tolower(liso.nucl) # a biblioteca seqinr opera com
# letras minúsculas, havendo a necessidade de conversão das maiúsculas
# obtidas pelo FASTA
# seqinr::count(liso.nucl2,1) # a mesma operação acima,
# mas com a biblioteca seqinr, e outro formato de chamada

# Outros cálculos
# seqinr:: count(liso.nucl2, 1 )
# seqinr::count(liso.nucl2,2) # teor de dinucleotídios
# seqinr::count(liso.nucl2,3) # teor de trinucleotídios
```

Outras manipulações da sequência, como o conteúdo de pares GC, gráfico da sequência de dinucleotídios, conversão da sequência de bases em uma sequência numérica e sua plotagem, e obtenção da sequência de bases complementar, por exemplo, podem ser obtidos por:

```
nucls <- table(liso.nucl)
GC <- 100 * (nucls[2] + nucls[3]) / (nucls[1] + nucls[2] + nucls[3] + nucls[4])
cat("percentual de conteúdo GC em lisozima de galinha: ", round(GC, 3))
```

```
percentual de conteúdo GC em lisozima de galinha:  59.014
```

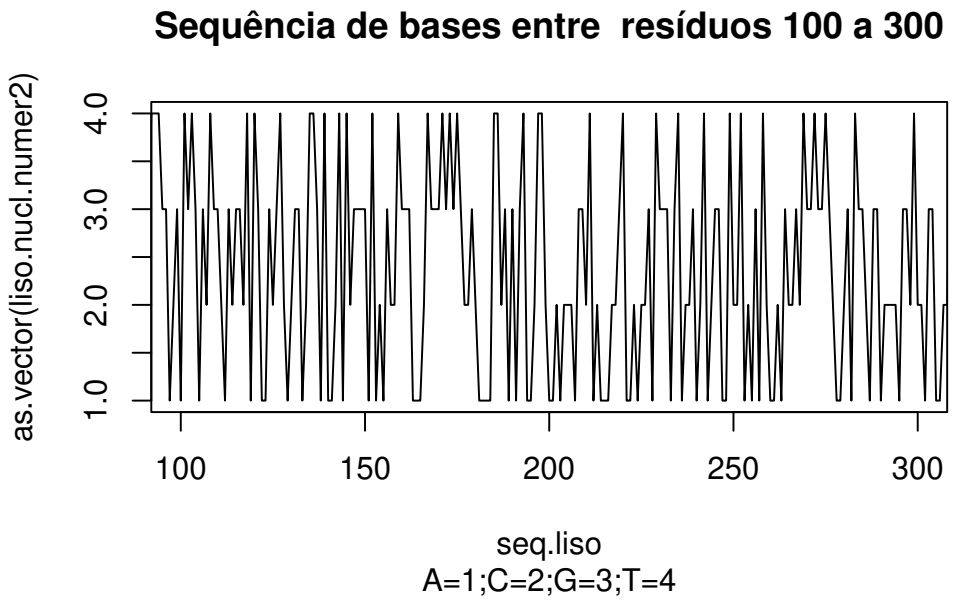
```
GC(liso.nucl) * 100 # o mesmo comando anterior, mas com a biblioteca seqinr
```

```
[1] 59.01361
```

```
# contag.liso <- count(liso.nucl2,2)
#
# barplot(sort(contag.liso)) # gráfico de barras do teor de dinucleotídios
# organizado por frequência

# Conversão de sequência nucleotídica em numérica
liso.nucl.numer <- gsub("T", "4", gsub(
  "G", "3",
  gsub("C", "2", gsub("A", "1", liso.nucl))
)) # substitui bases por valores
liso.nucl.numer2 <- as.numeric(liso.nucl.numer)
liso.nucl.numer2[1:100] # 100 primeiros valores da sequência
```

```
 [1] 3 2 1 3 4 2 2 2 3 2 4 3 4 3 4 3 4 1 2 3 1 2 1 2 4 3 3 2 1 1 2 1 4 3 1 3 3
[38] 4 2 4 4 4 3 2 4 1 1 4 2 4 4 4 3 3 4 3 2 4 4 4 3 2 4 4 2 2 4 3 2 2 2 2 4 3 3
[75] 2 4 3 2 4 2 4 3 3 3 3 1 1 1 3 4 2 4 4 4 3 3 1 2 3 1
```

```
# Obs: também pode ser obtido pelas funções s2n e n2s do pacote seqinr

seq.liso <- seq(1:length(liso.nucl))
plot(seq.liso, as.vector(liso.nucl.numer2),
  type = "l",
  xlim = c(100, 300), main = "Sequência de bases entre  resíduos 100 a 300",
  sub = "A=1;C=2;G=3;T=4"
)
```




```
# Obtenção de sequência complementar
comp.liso.nucl <- seqinr::comp(liso.nucl)
head(seqinr::c2s(comp.liso.nucl), 50) # apresenta os primeiros
```

```
[1] "cgtcagggcgacacacatgctgtgaccgttgtactccagaaacgattagaaccacgaaacgaaggacggggaccgacgagacccctttcagaaacctgcta
```

```
# 50 nucleotídios complementares
```

Sec 8.2 Termoestabilidade de DNA

Em relação à propriedades físico-químicas de ácidos nucleicos, é bem conhecida a relação entre a termoestabilidade de DNA duplex e o conteúdo de pares GC, tal como explicitado pela relação empírica (Creighton et al. 2010):

$$Tm = (81,5 + 16,6 * log(\frac{[Na^+]}{1 + 0,7 * [Na^+]}) + 41 * f_{GC} - \frac{500}{L} - 0,63\%_f) \quad (8.1)$$

Onde [Na^+] representa a concentração molar de sódio, f_{GC} a fração de pares GC da sequência, L seu comprimento, e $\%_f$ o teor de formamida.

Dessa forma é possível prever o valor de Tm ("melting temperature") que indexa a termoestabilidade de uma sequência polinucleotídica em função do teor salino. Ilustrando-se para uma comparação entre a sequência da lisozima de galinha e a humana (NCBI ref. NC_000012.12), na ausência de formamida:

```
# Comparação de curvas de desnaturação

# Para lisozima de galinha:
gc.teor <- seqinr::GC(liso.nucl) # teor de pares GC da lisozima
```

```
Na.conc <- seq(0.005, 0.2, 0.001) # concentração de NaCl, mmol/L

Tm.Na <- (81.5 + 16.6 * log10(Na.conc / (1 + 0.7 * Na.conc)) +
            41 * gc.teor - 500 / length(liso.nucl))
# valor de Tm para a de galinha

# Para a lisozima humana

liso.nucl.h <-
"AGCCTAGCACTCTGACCTAGCAGTCAACATGAAGGCTCTCATTGTTCTGGGGCTTGTCCTCCTTTCTGTT
ACGGTCCAGGGCAAGGTCTTTGAAAGGTGTGAGTTGGCCAGAACTCTGAAAAGATTGGGAATGGATGGCT
ACAGGGGAATCAGCCTAGCAAACTGTAAGTCTACTCTCCATAATTCCAGAGAATTAGCTACGTATGGAAC
AGACACTAGGAGAGAAGGAAGAAGAAGAAGGGGCTTTGAGTGAATAGATGTTTTATTTCTTTGTGGGTTT
GTATACTTACAATGGCTAAAAACATCAGTTTGGTTCTTTATAACCAGAGATACCCGATAAAGGAATACGG
GCATGGCAGGGGAAAATTCCATTCTAAGTAAAACAGGACCTGTTGTACTGTTCTAGTGCTAGGAAGTTTG
CTGGGTGCCTGAGATTCAATGGCACATGTAAGCTGACTGAAAGATACATTTGAGGACCTGGCAGAGCTCT
CTCAAGTCCTTGGTATGTGACTCCAGTTATTTCCCATTTTGAACTTGGGCTCTGAGAGCCTAGAGTGATG
CAGTATTTTTCTTGTCTTCAAGTCCCCTGCCGTGATGTGGGATTTTTATTTTTATTTTTATTTTATTTTA
TTTTATTTTTAAAGACAGTCTCACTGTGTGGCCCAGGCTGGAGTGCAGTGGCATGATCTCAGCTCACTGC
AACCTCTGCCTTCTGGGCTCAAGTGATTCTCGTGCTTCAGCCTTCTGAGTAGCTGTGACTACAGGTGTGT
ACCACCACACCCAGCTAATTTTTTGTATTTTCAGTACAGATGGGGTTTCACCATGTTGGCCAAGCTGGTC
TTGAACTCCTGGCCTCAAATGATCTGCCCACCTCAGCCTCCCAAAGTGGTAGGATTACAGGTGTGAACCA
CTGCACCCAGCCGACATGGGATTTTTAACAGTGATGTTTTTAAAGAATATATTGAATTCCCTACACAAGA
GCAGTAGGAACCTAGTTCCCTTCAGTCACTCTTTGTATAGGATCCCAGAAACTCAGCATGAAATGTTTTA
TTATTTTTATCTACTCTACTTGATTAACTATCTTTCATTTTCTCCCACACAATTCAAGATGTGCCATGAG
GAAAAGTTATTTTATAGTTTAGTACATAGTTGTCGATGTAATAATCTCTGTAGTTTTCAGATTGAATTCA
GACATTTCCCCTCAATAGCTATTTTTGAATGAATGAGTGAAGGGATGAAATCACGGAATAGTCTTGTTTT
CAAGATTCTAACTTGATATCCAAATTCACCTTTAGATATTATAAGAAAATTTCTATCAGAAAATCCTTAT
GTTTTTCTGATTAAAAAAAGCATTTTTCCATCAGCCTATGTATCTGCTATGAATTTACAAAATCTACTCA
ACAGCTCTGTTGATTTTTCTGTTCTTGGCTGAATGTTGCCTGAGGGATGGGAGCACGGGAAGGGTAAAAG
CAATGGAACAAACATGTATTTTAATATTTTAAAAGTATGTTATATTGTTCGTTGGTGTTACAAGATGATT
TGCATTACAAAAGGATTCTCTTACAAGTCCCTTATCTTAACACTAAAGTGCTAAGATATTTTATAAGTAA
ATCTTTATACTTATAAAACAAATCAGTAAAATAGAAGTAGCTAAGTAGAACTGATTTTGCTATAGAGTAT
AAGTCACTTAGTGTTGCTGTTTATTACTAAAAATAAGTTCTTTTCAGGGATGTGTTTGGCCAAATGGGAG
AGTGGTTACAACACACGAGCTACAAACTACAATGCTGGAGACAGAAGCACTGATTATGGGATATTTCAGA
TCAATAGCCGCTACTGGTGTAATGATGGCAAAACCCCAGGAGCAGTTAATGCCTGTCATTTATCCTGCAG
TGGTAAGACAAGCTAATATTTGACCAATCTGGTTATACTTACAAGAATTGAGACTCAATACAAATGAAAA
AGCCTTGAAAGGTTCATGAGGGACCTAGAAAAACTACATCTCAACTTCCAGAAAGTCATTATTATTTTCC
TCATAATTCCCTGAGTAAGAAATTAAAGAAGTGGTATCATAAAAGGTTGATGTTTTTTAATATACAGAAG
TTTCTGGAATGACCTATTAATTTACTGTCAATGGCCTTACTGATGCTTTGTCCAGAACAATGCCATTGCT
CCTGCTTACTTTGGGGAGGTTTTGGGATAATTTAGTTGTATGGTCCTTTTTCAATTGTTTTACTTTTTTT
TTTATGAAATGTTCTAAATGTATAGAAAATTAGAGACATTAGTATAATAAACAGCCATATGCCCATTATG
CACTTTAAAAGTTGTTAACATTTTGCCATAGTTGCTTCTTCTATGCCTTTTTTTTTTTTTTTTTTTTTTT
TTTTTTTTTGCTGAGAGTTTTTTGTTTGGTTTTGTTTTGTTTTATTTTGAGACAGGGTCTCCCTGTCCCC
AGGCTGTAGTGCAGTGGCACCATCACAGCTCACTGCAGCCTCAAGTGATCATCCCACCACAGCCTCCCAA
GTAGCTGGGACTACAGGTGTGCACCACCATGCCTGGCAAATTTTTGAAATTTTTAGTACAGGCAAATTCT
GTGTTGCCCAGGCTGGTCTTGAACTCCTGAGTTCAAGCAATCTTCCCACCTCAGCCTCCTTAAGTGCTGG
AATTACAGGCGTTAGCCACTGTACCTGGCTACTGCTGAGAGACTTTTAAGTGAATTAGGAACATGATGAT
ATTCCATTTCTAAATTCTTTAGTTTACATCTTCAAAAAATACAGTTCCTGTAGAATTATTATTGTAAATA
ACAAATTAACTTAAGGATTTATTTATTTGGAGTGAAACAAATATTTTACTGAACTCATAAAAATAGAAAT
ACCATGTGGAATCCTCAGTGTCAAAAATATTGCAGAAATCTTGCAAAGTTGATATTATTAAATTGTTAAA
TATTAAAATTCCCAATAAAGAACATTAATCTTATTTCTAAAATCCAGTTAATTAAAAAAATTTATATTAT
ATAATAATATTTGGTCATTAAATAAAAATTAGAAAATACAAATAAGAAAAATAACACCCATAATCTTACT
ACCCAGAGGTTTATAACCATGGGTAAATTCTGGTATATATTCTTCCAGAATGTATATCAATCATGTGTAT
GAATGTTAAATTATATCATACACATATAAACCCACATACAAACATGTAAATACTGTGTGCTTTTGCAAAA
ATTAAATTGTATTATACACACGGCTTTACAATTTGCTTCTTATCACACAAAATTATTTGCATGTCAGCAA
ATACAAATCGGTTTTTAATGATCTTTTGCTCCATTTTCCAGATGAGAAAAAAATACAAATCTGTATCATC
ATTTTAAAAGAATGACTAGAATTTTAATATATGAATATTCTATAATTTACTGATCCAATTGTTACTATTG
```

```
AGCACTTAGGTTGTTTCCATTTTTCCCTCATAAATTGCTATGAATAGCTTTTTGTATACATCTTTGGGTG
CATTTCTTATTTCTTTTGGATAAATTTTCAATAATAGAACTGCTGAGTAAAATATCACTAGGTGTTTTTT
TACAGTGTCTAGTGCAAAGAAGACCTTTAATCATTTTGTTAATACTTCCAGAGCTTCCAATGACTTTGGT
AAATGAAGAAAAAAATGCTTCATTTCATGCTGAATGGGAGAGAATGAAGAGAGTTTTCCCCAACAATTAC
ACATATATGGACTCATAGAAAATAATATCTTACCATTCTTTCCACAGCCTAACAGAAAAAAGCTGGCTAA
ACCTAAATTTAAAATAAAATATCTATTAAAGTTTTTATTCCTTACCACCTGTCTTTCAGCTTTGCTGCAA
GATAACATCGCTGATGCTGTAGCTTGTGCAAAGAGGGTTGTCCGTGATCCACAAGGCATTAGAGCATGGT
ATGTTTTAAGTGTTAAAAGGGAAAACTATCTTACTCTACTGTTGATATATACAATGAGAGCAGACTTTTA
AAGACCAAAGTATGCTAATGACACCTCAAAATTGCAGCTTTTGGCTTATGCTAAATGATGTATTACCTAC
ATCCTTGAAGAAACAATCTACTTTAACTGATCCAGAATCTTACTCTTTTACTCCTCAATTTATTTTAGGG
GATTTCTAGAGTTTTAAGATGCTTCACACTCTATCAGTTCCTTGTCATATCTTGAAATTCTTTTTAGAAT
AAGTAAGTGTGGGCCGGGCACAGTGGCTCACGCCTGTAATCCCAGCACTTTGGGAGACCGAGGCAGATGG
ATCACCTGAGGTCAGGAGTTCGAGACCAGCCTGCCTAACATGGCAAAACCCCATCTCCACTAAAAATACA
AAAAATTAGCTGGGTGTGGTGGCAGGTGCCTGTAATCCCAGCCACTCGGGAGGCTGAGGCAGGAGACTTG
CTTGAACCCGGGAGGTGGAGGTTGCAGAGGATTGCGCCATTGTACTTCAGCCTGGGCGACAGAGTGAGAC
TCTGTCTCAAATAAATACATAAAAAATAAATGTGGAATTCACTTTGCAGTTGCTGCTGTACAACGCACAT
TACTCAATCTTTATGTTCGGCATTCTATGCTCTACTGAGAAATTTGGGTAGGAGTGAAGTATTTTGTATA
CATATCTTCATTTAATAAATAGCAATAGCTGGGTCTATCTTACTATTTTATCTATTGATAAAATATTTTG
TTTCCCCAAGGAGTGCGAAGTATGTATATTACAATGAAGATATGTTTTAACCTTTCACCATTTGCTTCAT
CTTTTTCTACAGGGTGGCATGGAGAAATCGTTGTCAAAACAGAGATGTCCGTCAGTATGTTCAAGGTTGT
GGAGTGTAACTCCAGAATTTTCCTTCTTCAGCTCATTTTGTCTCTCTCACATTAAGGGAGTAGGAATTAA
GTGAAAGGTCACACTACCATTATTTCCCCTTCAAACAAATAATATTTTTACAGAAGCAGGAGCAAAATAT
GGCCTTTCTTCTAAGAGATATAATGTTCACTAATGTGGTTATTTTACATTAAGCCTACAACATTTTTCAG
TTTGCAAATAGAACTAATACTGGTGAAAATTTACCTAAAACCTTGGTTATCAAATACATCTCCAGTACAT
TCCGTTCTTTTTTTTTTTGAGACAGTCTCGCTCTGTCGCCCAGGCTGGAGTGCAGTGGCGCAATCTCGGC
TCACTGCAACCTCCACCTCCCGGGTTCACGCCATTCTCCTGCCTCAGCCTCCCGAGTAGCTGGGATTACG
GGCGCCCGCCACCACGCCCGGCTAATTTTTTGTATTTTTAGTAGAGACAGGGTTTCACCGTGTTAGCCAG
GATGGTCTCGATCTCCTGACCTTGTGATCCACCCACCTCGGCCTCCCAAAGTGCTGGGATTACAGGCGTG
AGCCACTGCGCCCGGCCACATTCAGTTCTTATCAAAGAAATAACCCAGACTTAATCTTGAATGATACGAT
TATGCCCAATATTAAGTAAAAAATATAAGAAAAGGTTATCTTAAATAGATCTTAGGCAAAATACCAGCTG
ATGAAGGCATCTGATGCCTTCATCTGTTCAGTCATCTCCAAAAACAGTAAAAATAACCACTTTTTGTTGG
GCAATATGAAATTTTTAAAGGAGTAGAATACCAAATGATAGAAACAGACTGCCTGAATTGAGAATTTTGA
TTTCTTAAAGTGTGTTTCTTTCTAAATTGCTGTTCCTTAATTTGATTAATTTAATTCATGTATTATGATT
AAATCTGAGGCAGATGAGCTTACAAGTATTGAAATAATTACTAATTAATCACAAATGTGAAGTTATGCAT
GATGTAAAAAATACAAACATTCTAATTAAAGGCTTTGCAACACA"

liso.nucl.h <- unlist(strsplit(liso.nucl.h, ""))
# converte sequência gênica de uma palavra em nucleotídios separados
liso.nucl.h <- liso.nucl.h[liso.nucl.h != "\n"]
# elimina a quebra de linha do resultado anterior

gc.teor.h <- seqinr::GC(liso.nucl.h) # teor de pares GC da lisozima hunana

Tm.Na.h <- (81.5 + 16.6 * log10(Na.conc / (1 + 0.7 * Na.conc)) +
              41 * gc.teor.h - 500 / length(liso.nucl.h))
# valor de Tm para a humana

# Curvas de simulação
plot(Na.conc, Tm.Na,
  type = "l", col = 2,
  xlab = "[Na+], M", ylab = "Tm, oC"
)
lines(Na.conc, Tm.Na.h, type = "l", col = 3)
legend(x = 0.13, y = 78, legend = c("galinha", "humana"),
       col = c(2, 3), cex = 1, lty = c(1, 2))
```

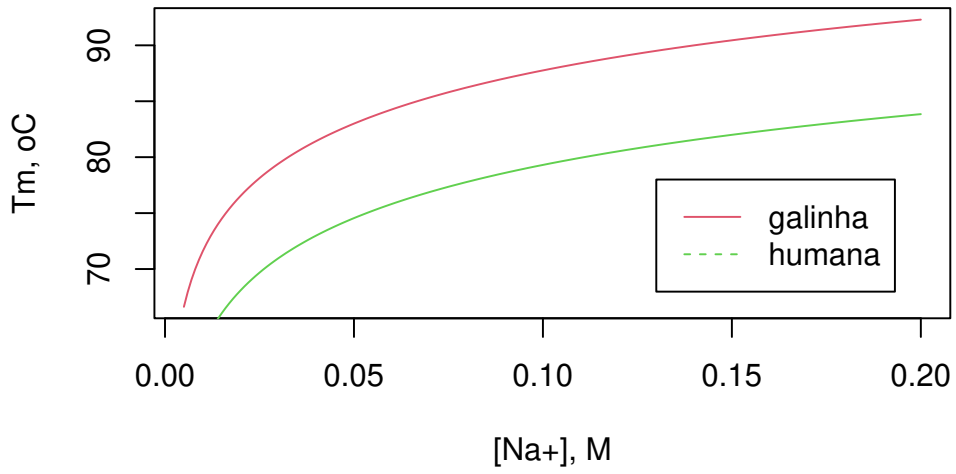


Figura 8.1: Comparação entre a curvas simuladas de Tm para a sequência nucleotídica da lisozima de galinha e lisozima humana, em função do teor de NaCl do meio.

Observe como a diferença no teor de GC tem efeito direto na termoestabilidade de fitas duplas de DNA. Uma observação: embora a faixa do valor de Tm relatado na literatura para a lisozima encontra-se em torno de 74°C, esse valor refere-se à desnaturação cooperativa da enzima em solução aquosa, e não ao desenovelamento de sua sequência gênica de DNA duplex.

Sec 9.1 Concentração micelar crítica (CMC)

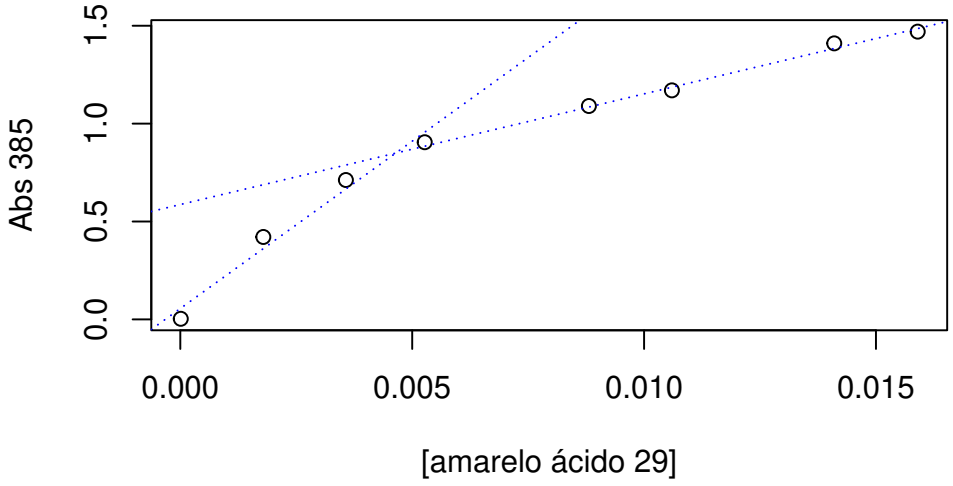
A concentração micelar crítica refere-se ao teor de um surfactante mínimo acima do qual esse pode reorganizar-se em micelas ou lipossomos. É bastante utilizado na caracterização de tais compostos, como biosurfactantes na indústria, e pode ser medido por diversas técnicas, incluindo tensão superficial, fluorescência de polarização, turbidimetria e absorção molecular (fotometria).

De modo geral a determinação de *cmc* é obtida pelo valor da concentração do surfactante no ponto de cruzamento de duas retas obtidas por ajuste linear dos dados em baixo e alto teor do analito, como segue:

$$y_1 = y_2 a_1 + b_1 * x = a_2 + b_2 * x a_1 - a_2 = b_2 * x - b_1 * x a_1 - a_2 = x(b_2 - b_1)x = \frac{a_1 - a_2}{b2 - b1} \tag{9.1}$$

O ponto de intersecção pode ser obtido manualmente, pelo comando *locator* já referido, ou de forma automática. Nesse caso, o exemplo do *R* que ilustra esse cálculo baseia-se em resultados de espectrofometria obtidos para um corante, o amarelo ácido 29 (Duff e Giles 1972).

```
# Determinação de concentração micelar crítica (CMC)

conc <- c(1e-5, 0.00179, 0.00357, 0.00527, 0.00881, 0.0106, 0.0141, 0.0159)
# teor do corante (mol/L)
A385 <- c(0.003, 0.421, 0.712, 0.905, 1.09, 1.17, 1.41, 1.47)
# absorbância em 385 nm

# Gráfico
plot(A385 ~ conc, xlab = "[amarelo ácido 29]", ylab = "Abs 385")

# Ajuste linear para 2 conjuntos de dados

# 1o. conjunto
linCmc1 <- lm(A385 ~ conc, subset = (conc < 0.007 & conc > 0))
# 1o. ajuste linear com limites
abline(linCmc1, col = "blue", lty = "dotted") # linha de regressão

# 2o. conjunto:
linCmc2 <- lm(A385 ~ conc, subset = (conc < 0.02 & conc > 0.007))
# 2o. ajuste linear com limites
abline(linCmc2, col = "blue", lty = "dotted") # linha de regressão
```




```
# Cálculo de CMC por intersecção das automática das duas retas:
cmc_auto <- abs((coef(linCmc2)[1] - coef(linCmc1)[1]) /
                  (coef(linCmc1)[2] - coef(linCmc2)[2]))
as.numeric(cmc_auto) # fornece o cmc em mol/L
```

```
[1] 0.004642773
```

O valor encontrado pelos autores foi de 0,004 mol/L.

Sec 9.2 Transporte em membranas e Teoria Quimiosmótica

O equilíbrio de transporte de solutos por membranas envolve um formalismo que abrange o potencial eletroquímico dos solutos envolvidos, suas concentrações (ou atividade), cargas, potenciais elétricos e volumes parciais em unidade molal. À despeito dessa complexidade, contudo, podemos ilustrar o transporte de íons H^+ de modo simplificado, seguindo-se a equação abaixo:

$$\Delta G_{transp} = 2.303(RT * log\frac{H^+_{in}}{H^+_{out}}) + z * F * \Delta\phi$$

Onde F representa a constante de Faraday, 96485 $J^{-1}V^{-1}mol^{-1}$ (também representado como 1 mol de elétrons), e z a carga do íon equanto que $\Delta\phi$ representa a variação de potencial elétrico, e ΔpH a variação do valor de pH, ambos obtidos por medições entre o lado interno (matriz mitocondrial) e externo (espaço intermembranas). $H^+{}_{in}$ e $H^+{}_{out}$ representam o teor de prótons do lado interno e externo da membrana, respectivamente.

Agora, considerando a carga unitária de H^+ e a definição para pH (-log H^+),

$$\Delta G_{transp} = \Delta\phi * F - 2.303RT * \Delta pH$$

Tangente ao transporte de solutos e íons por membranas celulares, é possível prever-se, por exemplo, o teor de ATP formado durante a fosforilação oxidativa que envolve o retorno de íons H^+ do espaço intermembranas à matriz mitocondrial. Ilustrando-se, considerando um valor de $\Delta\phi$ de 70 mV e um ΔpH de 1,4 para as medidas entre matriz e espaço intermembranas mitocondriais, prevê-se a obtenção de ATP pelas relações que seguem, considerando a energia de 31 kJ/mol de ATP :

```
# Teor previsto de ATP durante fosforilação oxidativa

R <- 8.341 # J/mol
T <- 298 # K
```

```
F <- 96485 # constante de Faraday
Dphi <- 70e-3 # variação de potencial elétrico in/out membranas
DpH <- -1.4 # variação de pH in/out membranas
DG_transp <- F * Dphi - 2.303 * R * T * DpH # equação de transporte
DG_transp_4 <- 4 * DG_transp # 4 mol de H+

# Considerando cada mol de ATP para 31 kJ/mol...
DG_transp_4 / 31e3
```

```
[1] 1.905559
```

Percebe-se, portanto, a produção de 2 mols de ATP nas condições explicitadas.

Sec 9.3 Proteínas de transporte em membranas

Enquanto prótons como H^+ são transportados em função de seu gradiente de concentração entre o lado interno e externo de membranas, outros compostos e solutos dependem de uma proteína transportadora, como glicose e ácido cítrico. Nesse caso, o transporte não é passivo, mas de *difusão facilitada*, e seu comportamento cinético pela membrana obedece o formalismo de *Michaelis-Menten* como segue.

$$v_{transp} = \frac{V_{max} * S_{out}}{K_{transp} + S_{out}}$$

Onde, analogamente, V_{max} representa a velocidade máxima (ou limite) de transporte do substrato, S_{out} o teor do substrato transportado, e K_{transp} a constante de dissociação do complexo proteína-substrato (ou concentração do substrato a meia saturação do transportador).

Enquanto a *Cinética* trata do *fluxo* de informações que envolvem um fenômeno, a *Termodinâmica* trabalha com as *forças* nele envolvidas. Essas forças denominadas *quantidades termodinâmicas* auxiliam a compreensão de variados fenômenos biológicos, tais como os equilíbrios elencados abaixo.

1. Estabilidade de biopolímeros;
2. Interação ligante-receptor;
3. Transporte de biomoléculas e íons;
4. Mudanças conformacionais em biomacromoléculas;
5. Associação de biopolímeros;
6. Transferência de elétrons em proteínas;
7. Combustão e síntese de biomoléculas;
8. Geração de energia metabólica.

Ainda que não caiba à *Termodinâmica* explicações sobre a teoria atômica, mecanismos moleculares ou taxas de reação, seu formalismo teórico permite avaliar as mudanças de energia (*entalpia, entropia, e energia de Gibbs*) que ocorrem entre o estado inicial e final de uma transformação. A partir dessas quantidades, é possível esboçar modelos mecanísticos das transformações envolvidas, baseados no conjunto empírico de observações similares reportadas.

A *equação de Gibbs* para o equilíbrio que descreve essas quantidades é:

$$\Delta G = \Delta H - T * \Delta S \tag{10.1}$$

Exemplifica-se para isso as tranformações nos valores de ΔH e ΔS que podem ser extraídos de uma transição conformacional que acompanha a desnaturação térmica de uma proteína (Cooper 2004).

Para isto é necessário determinar-se o valor de ΔG da *transição de fase*, o que pode ser realizado de diversas maneiras, e a partir de metodologia igualmente diversificada. Assim, por meio de medidas espectroscópicas (*absorção molecular, fluorescência, luminescência*), hidrodinâmicas (*viscosimetria*, *coeficiente de sedimentação*, *pressão osmótica*), eletroquímicas (*potenciometria, voltametria*), ou de atividade biológica, dentre muitas, é possível quantificar o parâmetro termodinâmico ΔG. Esse, por sua vez, pode ser extraído das relações que seguem, considerando-se uma *transição de dois estados*:

$$N \rightleftarrows D \tag{10.2}$$

$$K_{eq} = \frac{[D]}{[N]} \tag{10.3}$$

$$\Delta G = -R * T * ln(K_{eq}) \tag{10.4}$$

Onde K_{eq}, *[D]*, e *[N]* representam, respectivamente, a constante de equilíbrio de desnaturação da proteina, bem como as concentrações dessa na forma desnaturada e nativa.

Uma rápida observação da Equação 10.22 deixa claro tratar-se de uma função linear com a temperatura. Dessa forma é plausível imaginar um sistema no qual as quantidades termodinâmicas acima (K_{eq} e, consequentemente, ΔG) podem ser determinadas com variação da temperatura. Colocando em números:

Em 35^{o}C: ΔG_{desn} = ΔH_{desn} - 308 * ΔS_{desn} = +4,4 kJ mol^{-1}

Em 45^{o}C: ΔG_{desn} = ΔH_{desn} - 318 * ΔS_{desn} = -5,2 kJ mol^{-1}

Perceba que uma solução para o problema envolve a resolução sequencial das equações, subtraindo-se uma da outra para uniformizar uma incógnita (digamos, ΔS) que aplicada à outra equação resultará na segunda incógnita (no

caso, ΔH). Ainda que verossímil, esse procedimento é manual e perde valor se imaginarmos uma 3a. temperatura ensaiada para a desnaturação proteica em questão.

Outra solução, mais prática e computacional, envolve a resolução do *sistema de equações lineares*, tal como segue:

$$a_{11} * x_1 + a_{12} * x_2 = b_1 a_{21} * x_1 + a_{22} * x_2 = b_2 \tag{10.5}$$

Onde x_1 e x_2 representam, respectivamente, ΔH_{desn} e ΔS_{desn}.

Nesse caso, pode-se montar um *sistema matricial*, tal que:

$$a_{11} * x_1 + a_{12} * x_2 = b_1 a_{21} * x_1 + a_{22} * x_2 = b_2 \tag{10.6}$$

Ou seja,

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix},$$

$$x = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix},$$

$$b = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Resolve-se agora os valores de x (ou Δ's) linearmente:

$$A * x = b \tag{10.7}$$

Utilizando-se *álgebra matricial*, soluciona-se a Equação 10.7 para os valores de x:

$$x = A^{-1} * b \tag{10.8}$$

Por tratar-se de um *sistema de equações lineares*, essa solução tem em si a premissa de que os valores de ΔH e ΔS não variem na faixa de temperatura estudada.

Sec 10.1 Solução de sistema de equações lineares no R

Pra solucionar o problema da seção anterior pelo `R`, define-se inicialmente a *matriz* para A e a matriz para b tal que:

$$A = \begin{bmatrix} 1 & -308 \\ 1 & -318 \end{bmatrix},$$

$$b = \begin{bmatrix} +4,4 \\ -5,2 \end{bmatrix}$$

Assim,

```
# Definição de matrizes
A <- matrix(c(1, -308, 1, -318), ncol = 2, byrow = TRUE) # matriz A;
# o sinal negativo decorre da função possuir inclinação negativa
b <- matrix(c(4.4, -5.2), ncol = 1) # matriz b
```

Conforme representado na Equação 10.8, a solução matricial pode ser obtida pelo comando `solve`:

```
# Solução matricial para sistema linear
solve(A) %*% b #  # a operação %*% indica o produto escalar de dois
```

```
       [,1]
[1,] 300.08
[2,]   0.96
```

```
# vetores ("dot product")
```

Nesse caso, os parâmetros termodinâmicos encontrados foram $\Delta H_{desn} = 300$ kJ mol^{-1} e $\Delta S_{desn} = 960$ J mol^{-1}.

Observe a notação "%*%" para a multiplicação de duas matrizes na última linha do código. Trata-se de *multiplicação cruzada* ou *dot product* de duas matrizes. A multiplicação de matrizes é definida somente para duas matrizes dimensionalmente compatíveis em uma dada ordem. Essa implica que o número de colunas da 1a. matriz seja igual ao número de linhas da 2a. matriz. Nesse caso a matriz resultante terá o mesmo número de linhas da 1a. matriz e o mesmo número de colunas da 2a. matriz. Veja o exemplo:

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} * \begin{pmatrix} 1 & 2 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{pmatrix} \tag{10.9}$$

Outra observação também deve ser pontuada em relação à solução matricial de sistemas lineares. Uma reflexão rápida sobre a natureza linear da Equação 10.22 de Gibbs e de sua aplicação à solução de parâmetros termodinâmicos para o sistema de equações lineares acima sugere que poderíamos obter outros valores para ΔG a partir de outras temperaturas ensaiadas. Supondo que fossem, digamos, 5 ou 6 valores de T com seus respectivos valores de ΔG_{desn}, e reforçando a premissa de que os parâmetros ΔH_{desn} e ΔS_{desn} permanecessem constantes ao longo da faixa termal, poderíamos facilmente concluir tratar-se de uma relação linear de ΔG_{desn} em função de ΔH_{desn} T.

Dessa forma, tal como visto no capítulo Capítulo 5, poderíamos solucionar os parâmetros ΔH_{desn} e ΔSH_{desn} por *ajuste linear*. De fato, uma das expressões de *Van't Hoff* que definem esta relação linear é:

$$ln\, K_{eq} = -\frac{\Delta H}{R} * \frac{1}{T} + \frac{\Delta S}{R} \tag{10.10}$$

Outra consequência direta é a de que qualquer conjunto de pares de dados de variáveis dependente (y) e independente (x), e que exibem homogeneidade de variâncias e distribuição normal, tal como explicitado no Capítulo 5, pode também ser resolvido em seus parâmetros (*intercepto* e *inclinação*) com auxílio de álgebra matricial.

De fato, a solução matricial de ajuste linear pode ser obtida a partir da relação abaixo:

$$\beta = (X^T\, X)^{-1}\, X^T * y \tag{10.11}$$

Portanto, o ajuste linear ilustrado na Figura 5.6 do Capítulo 5 também pode ser efetuado com auxílio de matrizes, embora alguns indicadores estatísticos apresentados na tabela gerada pela função `lm` sejam extraídos por outras funções do algoritmo de cálculos matriciais/estatísticos. Na Equação 10.11 o termo entre parênteses envolve a operação de *inversão* da matriz. Em álgebra linear não existe a operação de divisão para matrizes, mas somente a multiplicação de uma matriz por um escalar ou pela *inversa* de outra. E mesmo assim, somente se tratar-se de uma *matriz quadrada*. Dessa forma o termo $(\mathrm{X}^T\ \mathrm{X})^{-1}$ só pode ser calculado com *inversão matricial*. No `R` essa ação é dada pelo comando `solve`.

Com dantes, é vital importância também que a matriz X contendo a variável independente seja criada com valores unitários à esquerda, tal como segue:

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ \dots & \dots \end{bmatrix}$$

Dessa forma, a solução do problema explicitado na Figura 5.6 do capítulo Capítulo 5, pode ser matricialmente resolvida como:

```
# Solução matricial para os parâmetros cinéticos de Lineweaver-Burk

# Repetindo os dados para as variáveis de Lineweaver-Burk
S <- seq(0.1, 1, length.out = 20) # gera uma sequência com 20 pontos
# entre 0 e 1 para valores de substrato
```

10

```
Vm <- 10
Km <- 0.5 # parâmetros cinéticos
set.seed(1500) # estabelece a mesma semente aleatória do gráfico direto
# de Michaelis-Menten, para reproducibilidade dos pontos
erro <- runif(20, 0, 1) # comando para erro uniforme (no. de pontos, min, max)
v <- Vm * S / (Km + S) + erro # equação de Michaelis-Menten
inv.S <- 1 / S # cria variáveis para o duplo-recíproco
inv.v <- 1 / v

# Criação das matrizes A e b
A2 <- matrix(c(rep(1, 20), inv.S), nrow = 20, byrow = FALSE) # cria matriz
# com valor unitário necessário antes da variável independente
b2 <- as.matrix(inv.v, nrow = 1, byrow = FALSE) # vetor b

# Solução matricial do ajuste linear
beta <- solve(t(A2) %*% A2) %*% t(A2) %*% b2
beta
```

```
           [,1]
[1,] 0.11363419
[2,] 0.03277244
```

Veja que os valores de intercepto (β 1) e inclinação (β 2) aproximaram-se dos encontrados com auxílio da função `lm` do R.

Sec 10.2 Matrizes e R

O emprego de matrizes na solução de problemas lineares e não lineares é bastante vasto. De fato, um ajuste linear é resolvido computacionalmente pelo uso de matrizes, mais do que por somatórias. Da mesma forma, alguns algoritmos para ajuste não linear também implementam álgebra matricial na solução de problemas (*Gauss-Newton*, *Levenberg-Marquadt*).

Dessa forma, é interessante uma rápida panorâmica no potencial de matrizes que o R possui.

```
# Algumas manipulações com matrizes

## Identificação de linhas e colunas
res <- matrix(c(-308, -318),
  nrow = 2, byrow = TRUE, # definição de matriz
  dimnames = list(c("Delta H", "Delta S"), "kJ/mol")
)
res
```

```
        kJ/mol
Delta H   -308
Delta S   -318
```

```
## Operações aritméticas
m1 <- matrix(c(1, 2, 3, 4), nrow = 2, byrow = T)
m2 <- matrix(c(4, 5, 6, 7), nrow = 2, byrow = T)
m1 + 5
```

```
     [,1] [,2]
[1,]    6    7
[2,]    8    9
```

```
m2 - 7 # soma ou subtração em escalar
```

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

```
m1^2
```

```
     [,1] [,2]
[1,]    1    4
[2,]    9   16
```

```
sin(m2) # potência ou trigonometria
```

```
           [,1]       [,2]
[1,] -0.7568025 -0.9589243
[2,] -0.2794155  0.6569866
```

```
m1 + m2 # soma de elementos em matrizes de igual dimensão
```

```
     [,1] [,2]
[1,]    5    7
[2,]    9   11
```

```
m1 * m2 # multiplicação de elementos em matrizes de igual dimensão
```

```
     [,1] [,2]
[1,]    4   10
[2,]   18   28
```

```
m1 %*% m2 # produto cruzado de matrizes ("dot product" de vetores)
```

```
     [,1] [,2]
[1,]   16   19
[2,]   36   43
```

```
det(m1) # determinante da matriz
```

```
[1] -2
```

```
t(m2) # transposição da matriz
```

```
     [,1] [,2]
[1,]    4    6
[2,]    5    7
```

```
diag(m1) # matriz diagonal
```

```
[1] 1 4
```

```
solve(m2) # inverso da matriz
```

```
     [,1] [,2]
[1,] -3.5  2.5
[2,]  3.0 -2.0
```

```
eigen(m1) # autovalor ("eingenvalue") e autovetor ("eigenvector") da matriz
```

```
eigen() decomposition
$values
[1]  5.3722813 -0.3722813

$vectors
           [,1]       [,2]
[1,] -0.4159736 -0.8245648
[2,] -0.9093767  0.5657675
```

O R também permite várias outras operações utilizadas em cálculo numérico e simbólico que utilizam matrizes, tais como as funções `kronecker` (multiplicação matricial), `svd` (*Single Decomposition Value*), `qr` (*Decomposição QR*), e `chol`(*Decomposição de Cholesky*).

10

Sec 10.3 Solução de parâmetros termodinâmicos de estabilidade enzimática

Parâmetros termodinâmicos, tais como os contidos na expressão de *Van't Hoff*, Equação 10.10, podem ser também obtidos simultaneamente por álgebra matricial. Exemplificandos-se, é comum em Biotecnologia a avaliação de parâmetros de termoestabilidade de enzimas submetidas a estresse térmico (como também químico, como pH, ureia ou uso de proteases). Isso é feito no intuito de se verificar, por exemplo, se uma enzima pode resistir a temperaturas elevadas de processos industriais, para comparar enzimas modificadas por mutação sítio-dirigida, ou para se avaliar o comportamento de enzimas associadas a patogenesias diversas. O formalismo dessa análise passa pela *Teoria das Colisões* de Arrhenius, bem como pela *Teoria do Estado de Transição* de Eyring, resultando no sistema linear de equações como o que segue:

$$\begin{cases} \Delta G^{\ddagger} = \Delta H^{\ddagger} - T * \Delta S^{\ddagger} \\ ln(\frac{kcat*h}{K_B*T}) = -\frac{1}{RT} * \Delta H^{\ddagger} + \frac{1}{R} * \Delta S^{\ddagger} \end{cases} \tag{10.12}$$

Onde os termos com ‡ simbolizam as variações de quantidades referentes à ativação (ou desativação) da enzima (*estado de transição do complexo ativado*), K_B representa a constante de Boltzmann (1,381x10^{-23} JK^{-1}), h a constante de Planck (6,686x10^{-34} J*s), e R a constante geral dos gases (8,314 JK^{-1} mol^{-1}). Nem sempre é possível convergir uma solução matricial pela simples utilização de multiplicação cruzada (*dot product*).

O trecho de código abaixo exemplifica essa tentativa, a partir dos dados publicados por Riaz e cols (Bhatti et al. 2007) abaixo, e considerando previamente as matrizes A e b em função dos parâmetros explicitados pelos autores:

$$\Delta G^{\ddagger} = 65920\, J\, mol^{-1} T = 328 K kcat = 217 s^{-1} \tag{10.13}$$

$$A = \begin{bmatrix} 1 & -328 \\ -3.67e-4 & 0.120 \end{bmatrix},$$

$$b = \begin{bmatrix} 65920 \\ -24.17 \end{bmatrix}$$

```
# Tentativa de solução matricial simples em dados publicados
# (Appl. Microbiol. Biotechnol, 73, 1290, 2007):

A <- matrix(c(1, -3.67e-4, -328, 0.120), nrow = 2, byrow = TRUE)
b <- matrix(c(65921, -24.17), nrow = 2)
solve(A) %*% b
```

```
             [,1]
[1,]    -21038593
[2,] -57505488910
```

Perceba a inconsistência para os parâmetros termodinâmicos resultantes. A solução matricial pelo comando `solve` também sofre de resolução para o problema, incorrendo em erro se executada, tal como no trecho que segue. Observe também a possibilidade distinta para a construção das matrizes.

```
Aframe <- data.frame(c(1, -3.67e-4), c(-328, 0.120))
A <- as.matrix(Aframe)
b <- as.matrix(c(65921, -24.17))
solve(t(A) %*% A) %*% t(A) %*% b
```

Para essas situações de maior complexidade pode ser de utilidade o emprego de pacotes do `R`, tal como o `rootSolve` já utilizado ou o `nleqslv` empregado anteriormente. Nesse sentido, a minimização de busca de raízes para o sistema de equações pode ser demonstrada a seguir:

```
# Minimização para raízes de sistema de equações termodinâmicas

library(rootSolve)
T <- 328
R <- 8.314
```

```
h <- 6.626e-34
kb <- 1.381e-23
kcat <- 217
DG <- -R * T * log((kcat * h) / (kb * T)) # 65920 J/mol
model <- function(x) c(x[1] - T * x[2] - DG, x[2] / R - x[1] / (R * T) -
                         log((kcat * h) / (kb * T)))
(ss <- multiroot(model, start = c(50000, 50000)))
```

```
$root
[1] 40843.50837   -76.45441

$f.root
[1]  2.110028e-09 -7.744916e-13

$iter
[1] 3

$estim.precis
[1] 1.055401e-09
```

Por esta *solução numérica* os valores encontrados para os parâmetros foram de Δ $H^{\ddagger}$ = 40,8 kJmol^{-1} e Δ $S^{\ddagger}$ = -76,5 Jmol^{-1}K^{-1}.

Comparando-se os valores, os autores encontraram Δ $H^{\ddagger}$ = 33,3 kJmol^{-1} e Δ $S^{\ddagger}$ = -99,8 Jmol^{-1}K^{-1}. Perceba a semelhança dos resultados obtidos pela minimização de raízes com os parâmetros encontrados pelos autores. A obtenção do valor de Δ $H^{\ddagger}$ para esses, contudo, foi obtida somente a partir da obtenção do valor experimental de *energia de ativação de Arrhenius (Ea)*, pela inclinação de um gráfico linearizado da taxa de reação, tal como segue:

$$k = A * e^{-Ea/RT} ln(k) = \frac{\Delta S^{\ddagger}}{R} - \frac{\Delta H^{\ddagger}}{R} * \frac{1}{T} \tag{10.14}$$

Ainda que se perceba uma significativa adequação dos valores obtidos pelos autores e em uma única temperatura, há que se ter cautela com o procedimento, pois minimizações costumam exigir boas sementes de inicialização do algoritmo para surtir bons resultados. Além disso, a natureza própria da relação termodinâmica entre uma taxa de reação (tal como *kcat*) e a variação de energia de Gibbs resultante ocorre em escala exponencial:

$$k = f(kcat) = e^{-\Delta G^{\ddagger}/RT} \tag{10.15}$$

Isso significa na prática que pequenas variações em Δ $G^{\ddagger}$ resultam em enormes variações no valor de k (no caso, de *kcat*). Por esta razão, diferenças mínimas no valor de Δ $G^{\ddagger}$ podem resultar em enormes diferenças em Δ $H^{\ddagger}$ e Δ $S^{\ddagger}$ para a solução do sistema linear. Exemplificando mais diretamente esse impacto, experimente alterar o valor de Δ $G^{\ddagger}$, arredondando-o:

```
require(rootSolve)
T <- 328
R <- 8.314
h <- 6.626e-34
kb <- 1.381e-23
kcat <- 217
DG <- 66000
model <- function(x) c(x[1] - T * x[2] - DG, x[2] / R - x[1] / (R * T) -
                         log((kcat * h) / (kb * T)))
(ss <- multiroot(model, start = c(50000, 50000)))
```

```
$root
[1] 51579492242   157254348

$f.root
[1] 177.50881195  -0.09422566

$iter
```

10

```
[1] 3

$estim.precis
[1] 88.80152
```

Sec 10.4 Entalpia De Reação Por Matrizes

Reações químicas são geralmente representadas pela equação que segue:

$$0 = \sum_{i=1}^{N} \nu_i B_i \tag{10.16}$$

Dessa forma, se observarmos as representações que acompanham as reações químicas, veremos que essas também constituem *funções lineares*, tal como nos exemplos abaixo:

$$2C_2H_2(g) + 5O_2(g) \rightleftarrows 4CO_2(g) + 2H_2O(l), \Delta_f H^o = -2600\, kJ/mol 2C_2H6(g) + 7O_2(g) \rightleftarrows 4CO_2(g) + 6H_2O(l),\ \Delta_f H^o = -3210 \tag{10.17}$$

E, se selecionarmos algumas reações que possuem relações entre si, tal como as apresentadas na Equação 10.17 acima, teremos então um *sistema de equações lineares*, passível de solução por álgebra matricial. Essa relação entre reações químicas que envolvem a formação de compostos refere-se à *Lei de Hess*.

Matematicamente a *Lei de Hess* pode ser expressa como:

$$\Delta_f H^o = \sum_{n=1}^{\infty} \nu \Delta_f H_P^o - \sum_{n=1}^{\infty} \nu \Delta_f H_R^o \tag{10.18}$$

Onde ν representa a *estequiometria* da reação, ou seja, o número de *mols* de cada composto/elemento, enquanto P e R referem-se à *Produto* e *Reagente*.

Retomando o exemplo da Equação 10.17, o que se deseja é obter o valor da *entalpia de reação* para o etano (C_2H_6), baseando-se nos valores de entalpia de reação das espécies correlatas (Khalil 2000). Há pelos três soluções possíveis, em que a entalpia de reação pode ser determinada pela *entalpia de ligação*, pela própria *entalpia de formação*, e pela *Lei de Hess*, essa passível de cálculo por matrizes.

Para isso, é necessário 1) compor as matrizes A e b, 2) calcular o vetor de coeficients *beta*, e 3) efetuar o produto escalar (%*%) de *beta* com uma matriz formada pelos valores de entalpia de formação. O racional para compor as matrizes envolve elencar cada composto com sua estequiometria de reação, e exige que para *reagentes* seja conferido valor negativo, enquanto que para produtos, valor positivo.

A tabela abaixo ilustra essa construção para o problema em questão.

```
library(knitr) # para gerar a tabela
comp <- c("C2H2", "O2", "CO2", "H2O", "C2H6", "H2") # elenco dos compostos
# envolvidos
rea1 <- c(-2, -5, +4, +2, 0, 0) # estequiometria (reação1)
rea2 <- c(0, -7, +4, +6, -2, 0) # estequiometria (reação2)
rea3 <- c(0, -0.5, 0, +1, 0, -1) # estequiometria (reação3)
incog <- c(-1, 0, 0, 0, +1, -2) # estequiometria da reação com entalpia
# desconhecida
tab_esteq <- data.frame(comp, rea1, rea2, rea3, incog) # dataframe com os
# resultados
colnames(tab_esteq) <- c("composto", "reação 1", "reação 2", "reação 3",
                         "etano") # nomeia as colunas
knitr::kable(tab_esteq, caption = "Estequiometria reacional para uma solução
             matricial de formação de etano (C2H6).", "pipe") # tabela
```

Tabela 10.1: Estequiometria reacional para uma solução matricial de formação de etano (C2H6).

composto	reação 1	reação 2	reação 3	etano
C2H2	-2	0	0.0	-1
O2	-5	-7	-0.5	0
CO2	4	4	0.0	0
H2O	2	6	1.0	0
C2H6	0	-2	0.0	1
H2	0	0	-1.0	-2

E o trecho de código que segue calcula o valor de ΔH_r^o para a formação de etano.

```
# Solução matricial para entalpia de formação
A <- matrix(c(-2, 0, 0, -5, -7, -0.5, 4, 4, 0, 2, 6, 1, 0, -2, 0, 0, 0, -1),
            nrow = 6, byrow = T) # cria matrix das reações com variação de
# entalpia conhecida
b <- matrix(c(-1, 0, 0, 0, 1, -2), nrow = 6, byrow = T) # cria matriz dos
# coeficientes estequiométricos da reação com variação de entalpia
# desconhecida
beta <- solve(t(A) %*% A) %*% t(A) %*% b # cálcula beta
energ <- matrix(c(-2600, -3210, -286), nrow = 1, byrow = T) # cria matriz
# com os valores de entalpia

etano <- energ %*% beta
cat("Valor para deltaHr etano: ", etano, " kJ/mol")
```

```
Valor para deltaHr etano:  -267  kJ/mol
```

Sec 10.5 Quantidades Termodinâmicas Por Ajuste Polinomial

Como visto nas seções anteriores deste capítulo e no de Capítulo 5, as *relações lineares* permitem a extração de parâmetros cinéticos ou termodinâmicos associados a fenômenos biofísico-químicos, tais como no estudo de associações de ligantes com biopolímeros, auto-associação de biomacromoléculas, cinética de enzimas, ou equilíbrio de estabilidade termodinâmica de biopolímeros. Referente à esse último, a Equação 10.10 ilustra a relação linear entre um parâmetro termodinâmico monitorado durante o experimento, tal como K_{eq} ou ΔG, e a temperatura (embora outros perturbantes poderiam igualmente viabilizar-se, tais como pH, teor de desnaturante, força iônica, etc).

Não obstante, não poderíamos utilizar as relações lineares de matrizes ou ajustes lineares para solucionar parâmetros quantitativos em situações que não repousassem no comportamento linear entre as variáveis, tal como referido pela Equação 10.10, por exemplo.

Ilustrando, a relação entre a temperatura e o valor para ΔG da auto-associação da apolipoproteína Apo A-II presente na lipoproteína HDL não exibe perfil linear, e pode ser obtida da literatura com auxílio do trecho de código que segue: (Waelbroeck, Van Obberghen, e De Meyts 1979).

```
# Dependência de T com deltaG para insulina e receptor

T <- c(5.29, 10.07, 15.23, 20.21, 25.11, 30.29, 37.39) + 273
# dados de temperatura, em graus Kelvin
dG <- c(11.74, 12.17, 12.46, 12.73, 12.88, 12.98, 13.13) * -1e3
# dados de -deltaG, em kcal/mol
plot(T, dG,
  xlab = "T, K", ylab = expression(paste(Delta, "G, kcal/mol"))
)
```

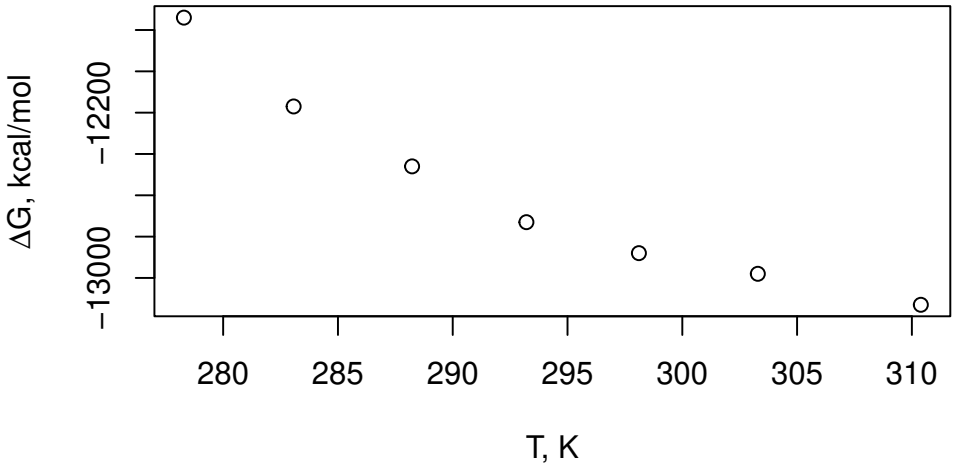


Figura 10.1: Dependência da temperatura com a variação de energia de Gibbs da interação de insulina com seu receptor.

Observa-se pela Figura 10.1 uma tendência parabólica entre a temperatura de ensaio e a variação de energia de Gibbs do processo. Dessa forma, pode-se ajustar um *polinômio* de grau 3 aos dados, tal como segue.

```
# Ajuste polinomial de parâmetros termodinâmicos

pol3 <- lm(dG ~ poly(T, 3, raw = TRUE)) # ajuste a polinômio de 3o. grau;
# "raw=TRUE" é essencial
# Alternativamente, pode-se também ajustar polinômios como
# pol3<-lm(dG ~ T + I(T^2)+I(T^3))

summary(pol3)
```

```
Call:
lm(formula = dG ~ poly(T, 3, raw = TRUE))

Residuals:
      1       2       3       4       5       6       7
  2.350 -13.432  26.731 -16.407  -8.577  12.164  -2.829

Coefficients:
                             Estimate Std. Error t value Pr(>|t|)
(Intercept)                 8.970e+05  2.449e+05   3.663   0.0352 *
poly(T, 3, raw = TRUE)1 -8.836e+03  2.498e+03  -3.537   0.0385 *
poly(T, 3, raw = TRUE)2  2.866e+01  8.492e+00   3.375   0.0433 *
poly(T, 3, raw = TRUE)3 -3.105e-02  9.613e-03  -3.230   0.0482 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 21.6 on 3 degrees of freedom
Multiple R-squared:  0.999, Adjusted R-squared:  0.9981
F-statistic:  1045 on 3 and 3 DF,  p-value: 5.02e-05
```

```
plot(T, dG,
  xlab = "T, K", ylab = expression(paste(Delta, "G, kcal/mol"))
  # gráfico de T x deltaG
```

```
)
lines(T, fitted(pol3), col = "red") # curva ajustada sobre os dados
```

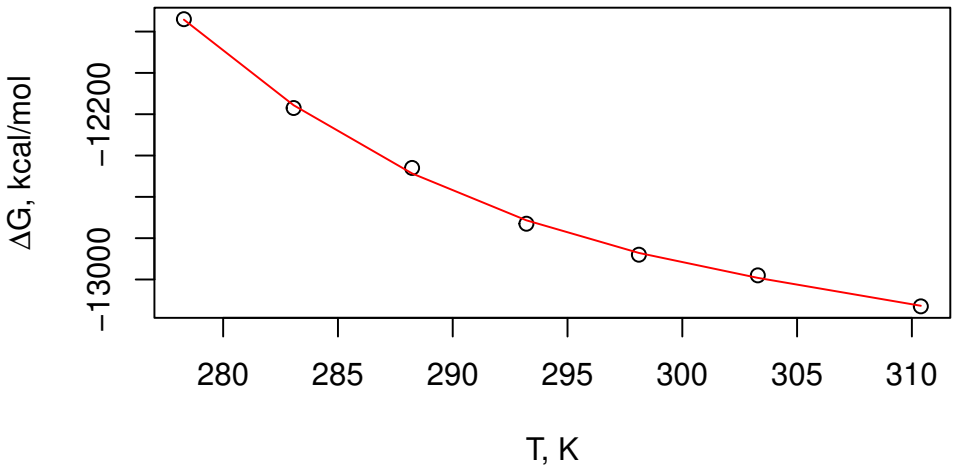


Figura 10.2: Ajuste polinomial sobre os dados de variação de energia de Gibbs da interação de insulina com seu receptor.

Aqui vale ressaltar que a obtenção dos parâmetros de um polinômio também pode validar-se com auxílio da álgebra linear (matrizes). Exemplificando, construa a *matriz A* dos valores de temperatura, e a *matriz b* dos valores de ΔG:

$$A = \begin{bmatrix} 1 & 278.29 \\ 1 & 283.07 \\ 1 & 288.23 \\ 1 & 293.21 \\ 1 & 298.11 \\ 1 & 303.29 \\ 1 & 310.39 \end{bmatrix},$$

$$b = \begin{bmatrix} -11740 \\ -12170 \\ -12460 \\ -12730 \\ -12880 \\ -12980 \\ -13130 \end{bmatrix}$$

Nesse caso, a operação matricial leva em conta a conversão da matriz A da variável independente numa *matriz de alternância*, também conhecida como *matriz de Vandermonde*. Uma matriz de Vandermonde se apresenta como abaixo:

$$matriz\, V = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 & ... \\ 1 & x_2 & x_2^2 & x_2^3 & ... \\ 1 & x_3 & x_3^2 & x_3^3 & ... \\ 1 & ... & ... & ... & ... \end{bmatrix},$$

Uma aparente limitação desse procedimento é que o ajuste deve ser realizado com poucos pontos experimentais, já que o termo exponencial cresce com o número de pontos. Por outro lado, a solução matricial contorna a necessidade

em se obter somatórias estatísticas das variáveis. O R possui um pacote para automatizar essa transformação, `matrixcalc`, exemplificado no trecho de código abaixo:

Agora basta aplicar a mesma relação matricial da Equação 10.8, no caso, para quatro pontos intercalados dos dados experimentais acima, e portanto produzindo um polinômio de 4o grau:

```
# Ajuste polinomial de 4o. grau para parâmetros termodinâmicos

T <- c(5.29, 15.23, 25.11, 37.39) + 273 # dados de temperatura,
# em graus Kelvin
dG <- c(11.74, 12.46, 12.88, 13.13) * -1e3 #

library(matrixcalc)
# Criação das matrizes A (Vandermonde) e b
b <- as.matrix(dG, nrow = 4, byrow = TRUE) # vetor b
A <- vandermonde.matrix(alpha = T, n = 4)
A # função para criar matriz de alternância (Vandermonde)
```

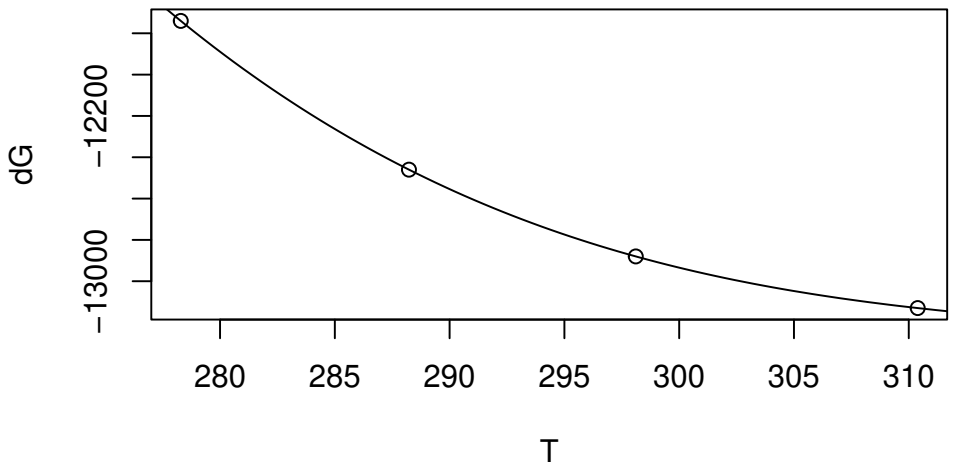
```
     [,1]   [,2]     [,3]     [,4]
[1,]    1 278.29 77445.32 21552259
[2,]    1 288.23 83076.53 23945149
[3,]    1 298.11 88869.57 26492908
[4,]    1 310.39 96341.95 29903579
```

```
sol.vnd <- solve(A) %*% b
sol.vnd # coeficientes polinomiais (4o. grau)
```

```
              [,1]
[1,]  5.095658e+05
[2,] -4.886799e+03
[3,]  1.525183e+01
[4,] -1.589352e-02
```

Para a plotagem dos dados, basta converter os coeficientes polinomiais acima na expressão polinomial resultante, o que pode ser realizado com o pacote `polynom`:

```
library(polynom) # converte vetor de coeficientes em polinômio simbólico
p <- as.polynomial(sol.vnd)
p2 <- as.function(p) # permite converter o polinômio pra função curve
plot(T, dG)
curve(p2, from = 273, to = 315, add = TRUE) # curva polinomial suave
```



Embora o ajuste polinomial, quer pela função própria do R (lm) como pela solução matricial acima, revele boa adesão do modelo aos dados experimentais, como representado pela Figura 10.2 e pela tabela de resultados acima, não há correlação de parâmetros termodinâmicos, posto tratar-se de um modelo matemático empírico, e não analítico para o sistema.

Entretanto, é possível obter uma boa aproximação das quantidades ΔH (entalpia), ΔS (entropia) e ΔCp (capacidade calorífica) que modelam fenomenologicamente o comportamento termodinâmico em determinada temperatura, por relações próprias entre essas quantidades (Edelhoch e Osborne Jr 1976).

$$\Delta S = -(\frac{\partial \Delta G}{\partial T})_p \tag{10.19}$$

Em suma, a variação de entropia pode ser definida como o gradiente da variação de energia de Gibbs com a temperatura. Outra forma de dizê-lo seria afirmar então que a variação de entropia pode ser também definida como a primeira derivada daquela relação, a qual pode ser definida empiricamente por:

$$\Delta G = a + bT + cT^2 + dT^3 \tag{10.20}$$

Assim, o valor de ΔS pode ser obtido pela primeira derivada da relação empírica acima (Equação 10.20):

$$\Delta S = -(\frac{\partial \Delta G}{\partial T})_p = -b - 2cT - 3dT^2 \tag{10.21}$$

O valor de ΔH, por sua vez, pode agora ser extraído da Equação 10.22 aqui repetida, juntamente com a Equação 10.20:

$$\Delta G = \Delta H - T * \Delta S \tag{10.22}$$

$$\Delta H = \Delta G + T * \Delta S \tag{10.23}$$

Aplicando-se as equações empíricas para os parâmetros termodinâmicos acima:

$$\Delta H = (a + bT + cT^2 + dT^3) + T(-b - 2cT - 3dT^2) \tag{10.24}$$

$$\Delta H = a - cT^2 - 2dT^3 \tag{10.25}$$

De forma similar, a *capacidade calorífica* em pressão constante pode ser definida como:

$$\Delta Cp = -(\frac{\partial \Delta H}{\partial T})_p \tag{10.26}$$

Ou seja, o valor de ΔCp pode ser aproximado pela primeira derivada de ΔH (Equação 10.24) sobre T. Ou seja:

$$\Delta Cp = -2cT - 6dT^2 \tag{10.27}$$

Ainda que constituam aproximações, os parâmetros termodinâmicos assim obtidos refletem a possibilidade de se descrever um fenômeno, tal como a interação de insulina com seu receptor (Waelbroeck, Van Obberghen, e De Meyts (1979)), sob a óptica de ligações fracas preponderantes, como ligações de hidrogênio, forças de van der Waals, efeito salino, interações eletrostáticas e efeito hidrofóbico (Ross e Subramanian 1981), apenas monitorando-se uma constante de equilíbrio com a temperatura. O trecho de código abaixo soluciona os parâmetros termodinâmicos para a complexação de insulina a seu receptor em 25^{o}C pelo método descrito.

```
# Solução polinomial de parâmetros termodinâmicos para interação de
# insulina com receptor

T <- c(5.29, 10.07, 15.23, 20.21, 25.11, 30.29, 37.39) +
  273 # dados de temperatura, em graus Kelvin
dG <- c(11.74, 12.17, 12.46, 12.73, 12.88, 12.98, 13.13) *
```

```
  -1e3 # dados de -deltaG, em kcal/mol

# Ajuste a polinômio de 2o. grau
pol3 <- lm(dG ~ poly(T, 3, raw = TRUE)) # ajuste a polinômio de 3o. grau

Tref <- 298 # temperatura de referência, em graus Kelvin

# Cálculos
dG <- coef(pol3)[1] + coef(pol3)[2] * Tref + coef(pol3)[3] *
  Tref^2 + coef(pol3)[4] * Tref^3 # deltaG
dS <- -coef(pol3)[2] - 2 * coef(pol3)[3] * Tref - 3 * coef(pol3)[4] *
  Tref^2 # deltaS
dH <- coef(pol3)[1] - coef(pol3)[3] * Tref^2 - 2 * coef(pol3)[4] *
  Tref^3 # deltaH
dCp <- -2 * coef(pol3)[3] * Tref - 6 * coef(pol3)[4] * Tref^2 # deltaCp

# Parâmetros em 298 K
cat("valor de deltaG: ", dG, "cal/mol", "\n")
```

```
valor de deltaG:  -12868.43 cal/mol
```

```
cat("Valor de deltaS: ", dS, "cal/mol/K", "\n")
```

```
Valor de deltaS:  27.29257 cal/mol/K
```

```
cat("Valor de deltaH: ", dH, "cal/mol", "\n")
```

```
Valor de deltaH:  -4735.248 cal/mol
```

```
cat("valor de deltaCp: ", dCp, "cal/mol/K", "\n")
```

```
valor de deltaCp:  -537.5956 cal/mol/K
```

Os valores encontrados para a interação são bem próximos dos reportados pelos autores em 25^{o}C (Waelbroeck, Van Obberghen, e De Meyts (1979)), embora esses tenham empregado um ajuste com polinômio de 2o. grau. Se você alterar o trecho de código acima para um polinômio de mesmo grau e omitir os termos das equações termodinâmicas que evidenciam o coeficiente d, deverá observar um valor de ΔCp de -735 kcal/mol, bem similar ao reportado de -766 kcal/mol.

Sec 10.6 Estabilidade Termodinâmica de Biopolímeros

A estabilidade termodinâmica de proteínas, enzimas e ácidos nucleicos é imprescindível do estudo de novos biopolímeros engenheirados, matrizes complexas modificadas (plasma artificial, por ex), bem como da pesquisa de candidatos à fármacos e medicamentos. Em linhas gerais, considera-se o biopolímero sob avaliação num *modelo de dois estados*, *nativo* e *desnaturado*, tal como apresentado na Equação 10.3 deste capítulo.

No entanto, como torna-se complexa a determinação experimental das concentrações *[N]* e *[D]*, busca-se obter uma relação entre elas, especificamente, sua fração, tal que:

$$f_D + f_N = 1 \tag{10.28}$$

O que resulta em:

$$f_N = 1 - f_D \tag{10.29}$$

Dessa forma, um sinal experimental S obtido na presença de frações N e D em uma amostra de biopolímero pode ser representado como:

$$S = f_N * S_N + f_D * S_D \tag{10.30}$$

Substituindo-se Equação 10.29 em Equação 10.30 obtém-se:

$$f_D = \frac{S_i - S_N}{S_D - S_N} \tag{10.31}$$

Onde *Si* representa o sinal no ponto *i*.

Dessa forma, mesmo que as concentrações *[N]* e *[D]* não sejam diretamente obtidas, suas frações podem ser recuperadas a partir do sinal obtido de ensaios de desnaturação frente a diversos perturbantes, tais como temperatura, pH, sais ou reagentes desnaturantes (ureia, cloreto de guanidina, por ex).

Dessa forma, também pode ser recuperado o valor da constante termodinâmica de equilíbrio de desnaturação K_D, tal que:

$$K_D = \frac{[D]}{[N]} = \frac{f_D}{f_N} \tag{10.32}$$

Inserindo-se Equação 10.29 em Equação 10.32, obtém-se:

$$K_D = \frac{f_D}{1 - f_D} \tag{10.33}$$

E, portanto,

$$\Delta G_D = -RT * ln\ K_D \tag{10.34}$$

Dessa forma, é possível avaliar a estabilidade termodinâmica de um biopolímero por sua *curva de estabilidade*, contrastando-se o *perturbante* contra o valor de ΔG_D obtido pelos procedimentos acima.

Analiticamente, uma curva de estabilidade pode ser gerada a partir dos parâmetros constantes na *equação integrada de Gibbs-Helmholtz* (LiCata e Liu (2011)):

$$\Delta G(T) = \Delta H_m(\frac{Tm - T}{Tm}) - \Delta Cp[Tm - T(1 - ln\ \frac{Tm}{T})]) \tag{10.35}$$

Assim, pode-se ilustrar uma curva de estabilidade pelo trecho de código que segue:

```
# Curva de desnaturação para proteína

Tm <- 85
dHm <- 180
dCp <- 3
x <- 0:80

curve(dHm * (1 - x / Tm) + dCp * ((x - Tm - x * log(x / Tm)))
      , xlim = c(0, 80)) # Nicholson1996; Sholz2009
```

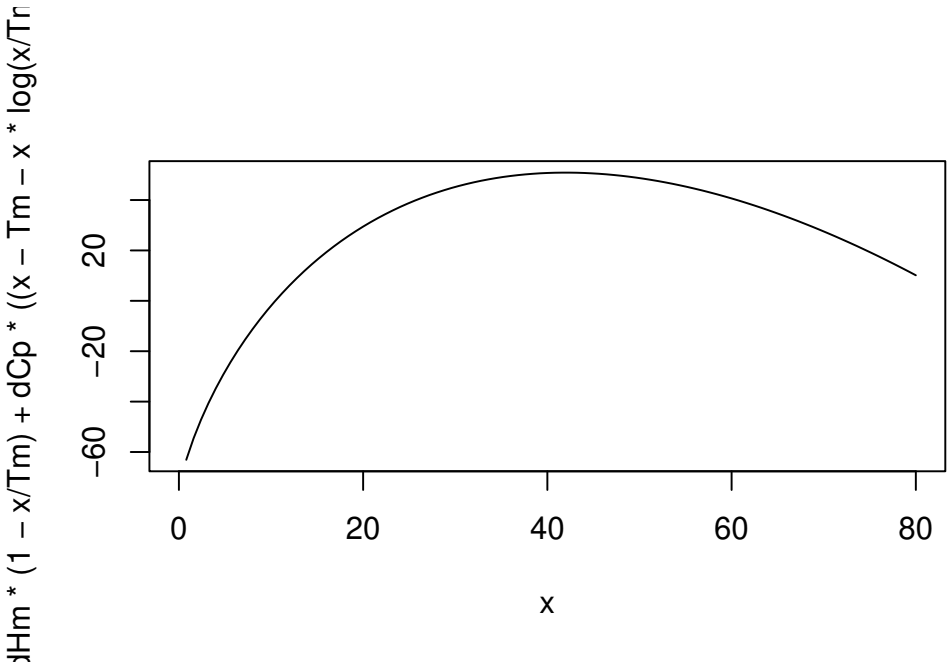


Figura 10.3: Curva de estabilidade simulada para a desnaturação de uma proteína. Tm = 75oC, DeltaHm = 180 kcal/mol, e DeltaCp = 3 kcal/mol/K.

Sec 10.7 Relação Quantitativa Estrutura-Função (QSAR) e Ajuste Multilinear

Os mesmos procedimentos empregados à solução de problemas matriciais, especificamente junto à Equação 10.11, podem acomodar também uma análise de *QSAR (Quantitaive Structure-Function Relationship)* de interesse. Por exemplo, derivados de benzodiazepinona (TIBO) são conhecidos inibir a transcriptase reversa (Tong et al. 2018), enzima que catalisa a conversão de RNA em DNA viral na síndrome de deficiência imunológica adquirida (SIDA). Nesse sentido, observações derivadas do estudo de QSAR podem contribuir para o desenho de potenciais inibidores da transcriptase do HIV. Assim, Tong e cols. propuseram um modelo preditivo multilinear relacionando algumas propriedades de análogos de TIBO com atividade biológica, como segue:

$$pIC_{50} = x_0 + x_1 * S + x_2 * W$$

{#eq:eqTIBO}

Onde pIC_{50} representa a atividade biológica aferida (*-log* IC_{50}), S indexa valores de solubilidade, e W refere-se a parâmetro de largura do primeiro átomo do grupo substituinte. Esses dados são tabelados abaixo:

```
# Dados de tabulação para QSAR

grupo <- c("H", "Cl", "SCH3", "OCH3", "CN", "CHO", "Br", "CH3", "CCH")
# grupos substituintes em TIBO
S <- c(3.53, 4.24, 4.09, 3.45, 2.96, 2.89, 4.39, 4.03, 3.8) # parâmetro de solubilidade
W <- c(1, 1.8, 1.7, 1.35, 1.6, 1.6, 1.95, 1.6, 1.6)
# parâmetro de largura de grupo
pIC50 <- c(7.36, 8.37, 8.3, 7.47, 7.25, 6.73, 8.52, 7.87, 7.53)
# atividade biológica de TIBO

tab.tibo <- data.frame(grupo, S, W, pIC50)
knitr::kable(tab.tibo, caption = "dados multivariáveis de atividade
             biológica de TIBO e parâmetros  preditivos.", "pipe") # tabela
```

Tabela 10.2: dados multivariáveis de atividade biológica de TIBO e parâmetros preditivos.

grupo	S	W	pIC50
H	3.53	1.00	7.36
Cl	4.24	1.80	8.37
SCH3	4.09	1.70	8.30
OCH3	3.45	1.35	7.47
CN	2.96	1.60	7.25
CHO	2.89	1.60	6.73
Br	4.39	1.95	8.52
CH3	4.03	1.60	7.87
CCH	3.80	1.60	7.53

Observe que existem duas variáveis preditoras e uma variável dependente, e cuja solução pode ser encontrada por *ajuste multilinear ou linear múltiplo*. O `R` permite fazê-lo por ao menos duas formas: função interna de ajuste linear (`lm`) ou álgebra matricial.

10.7.1 Ajuste linear múltiplo por função `lm`:

De forma simplificada, pode-se obter a expressão multivariável que define a relação das quantidades preditivas W e S com a atividade biológica de TIBO por:

```
# Ajuste multilinear em QSAR

lm.tibo <- lm(tab.tibo$pIC50 ~ tab.tibo$S + tab.tibo$W)
# comando para ajuste multilinear;

# Alternativamente,
# lm.tibo <- lm(cbind(S,W)~pIC50)
summary(lm.tibo)
```

```
Call:
lm(formula = tab.tibo$pIC50 ~ tab.tibo$S + tab.tibo$W)

Residuals:
     Min       1Q   Median       3Q      Max
-0.27636 -0.15649  0.02922  0.08911  0.24761

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.5903     0.5435   6.606 0.000579 ***
tab.tibo$S    0.9571     0.1519   6.300 0.000746 ***
tab.tibo$W    0.3619     0.3020   1.199 0.275888
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2045 on 6 degrees of freedom
Multiple R-squared:  0.912, Adjusted R-squared:  0.8826
F-statistic: 31.07 on 2 and 6 DF,  p-value: 0.0006826
```

Observe a forma mais direta de se atribuir variáveis já declaradas para uma função do `R` (*dataframe $ vetor). Essa é maneira mais simples, pois independe de pacotes extras (como o `dplyr`), embora seja menos legível.

Expressando-se os resultados na função linear múltipla:

$$y = 5,75 + 0,14 * S + 0,95 * W \tag{10.36}$$

10

10.7.2 Ajuste linear múltiplo por matrizes:

Os coeficientes *beta* obtidos acima podem ser também buscados por álgebra linear, utilizando-se a matriz b de atividade biológica e a matriz A contendo as variáveis independentes, essa também elaborada com valores unitários à esquerda, como antes, seguido da aplicação da Equação 10.11:

$$X = \begin{bmatrix} 1 & S_1 & W_1 \\ 1 & S_2 & W_2 \\ 1 & S_3 & W_3 \\ ... & ... & \end{bmatrix}$$

```
# Criação das matrizes A e b
A <- matrix(c(rep(1, 9), S, W), nrow = 9, byrow = FALSE)
# cria matriz com valor unitário necessário antes da variável independente
b <- as.matrix(pIC50, nrow = 1, byrow = FALSE) # vetor b

# Solução matricial do ajuste linear
beta <- solve(t(A) %*% A) %*% t(A) %*% b
beta
```

```
          [,1]
[1,] 3.5902556
[2,] 0.9571092
[3,] 0.3619292
```

Veja que os valores para os coeficientes são coincidentes. Na prática o procedimento de ajuste linear múltiplo pode ser utilizado, como no exemplo acima, para a predição de uma resposta (como pIC_{50}) em função de variáveis preditoras (como S e W).

Esse procedimento matricial multilinear também pode ser aplicado em outros tipo de análise multivariável, como *experimento fatorial* e *metodologia de superfície de resposta*. Isso decorre da própria natureza desses sistemas, quando lineares. Veja as aplicações abaixo. Mesmo para *metodologia de superfície de resposta quadrática* (onde os parâmetros variam com o quadrado das variáveis preditoras), também é possível a solução matricial (Equação 10.11).

$$y = b_0 + b_1 * x,\ ajuste\ linear y = b_0 + b_1 * x_1 + b_2 * x_2 + ... + b_n * x_n,\ ajuste\, multilinear y = b_0 + b_1 * x_1 + b_2 * x_2 + ... + b_n * x_n \quad (10.37)$$

Via de regra, todas as aplicações listadas em Equação 10.37 podem ser solucionadas com auxílio das relações matriciais das equações Equação 10.11 (ajustes linear, multilinear e superíficie de resposta linear) ou Equação 10.7 (planejamentos fatoriais). A seguir são exemplificadas duas situações dessa natureza.

10.7.3 Uma palavra sobre matrizes e aplicações

Como já percebido, o emprego de matrizes extende-se a situações diversas, não necessariamente de cunho Bioquímico ou Biofísico, ajustes de dados, Lei de Hess ou Lambert-Beer, como também a construção de um cardápio (matriz de ingredientes de cada prato por preço de cada ingrediente), ou o gasto calórico diário em atividade física (matriz de carga horária semanal de cada atividade por matriz de gasto calórico da atividade). De fato, são ferramentas utilizadas em Bioinformática, Economia, Ecologia, Engenharia, e tantas outras áreas, satisfeitas as relações básicas entre as variáveis em estudo. Essas condições, por sua vez, nada representam além do que *a somatória de produtos*, em que esses se manifestam por quantidades declaradas no problema. Assim, uma pequena representação da afirmação acima poderia ilutrar-se singelamente como:

$$y = \sum_{n=1}^{\infty} (x_1 * x_2) \quad (10.38)$$

Sec 11.1 Rendimento de Reação & Planejamento Fatorial 2^2

O *experimento fatorial* mais simples é o que se avalia a *resposta* de um experimento (rendimento, por ex) em que se variam dois *fatores* (temperatura e tipo de catalisador, por ex) em dois *níveis* cada (baixo e alto). A tabela elaborada no trecho abaixo a partir dos dados explicitados por Neto e cols (Neto, Scarminio, e Bruns 2010) exemplifica essa situação.

```
#  Planejamento fatorial

# Dados do experimento
temp <- c(40, 60, 40, 60)
catalis <- c("A", "A", "B", "B")
rendim <- c(59, 90, 54, 68)

# Tabela do planejamento fatorial

tab.fat <- data.frame(temp, catalis, rendim)
knitr::kable(tab.fat, caption = "Dados de experimento
             fatorial (Neto e cols, 2010).", "pipe") # tabela
```

Tabela 11.1: Dados de experimento fatorial (Neto e cols, 2010).

temp	catalis	rendim
40	A	59
60	A	90
40	B	54
60	B	68

Para conduzir a análise matricial dos dados é necessário converter a tabela de variáveis independentes (preditoras) em uma *matriz de planejamento codificada*, na qual valores altos (*nível superior*) são representados por *+1* e valores baixos (*nível inferior*) por *-1*. Além disso, também é necessário atribuir-se *+1* à 1a. coluna, e produzir uma 4a. coluna contendo o produto dos preditores codificados. Exemplificando, para temperatura a 40 (-1) e catalisador B (+1), a linha conterá o produto -1. Essa matriz final 4x4 é denominada *matriz de coeficientes de contraste*:

$$X = \begin{bmatrix} 1 & -1 & -1 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Além disso, é necessário dividir a o resultado da operação matricial por um vetor escalar específico. Resumindo: elabora-se a matrix X codificada dos preditores, a matriz Y da resposta, aplica-se a Equação 10.7, e divide-se o resultado por um vetor característico do planejamento fatorial 2^2 (*c(4,2,2,2)*). O trecho de código abaixo soluciona o problema levantado:

```
# Criação da matriz de planejamento codificada
X <- matrix(c(1, -1, -1, 1, 1, 1, -1, -1, 1, -1, 1, -1, 1, 1, 1, 1),
            nrow = 4, byrow = TRUE)
# Criação da matriz de rendimento
```

```
Y <- as.matrix(rendim)
# Determinação dos coeficientes beta:
beta <- t(X) %*% Y / c(4, 2, 2, 2)
beta
```

```
       [,1]
[1,]  67.75
[2,]  22.50
[3,] -13.50
[4,]  -8.50
```

Dessa forma, interpreta-se o resultado como:

$$\begin{pmatrix} M \\ T \\ C \\ TC \end{pmatrix} = \begin{pmatrix} 67,75 \\ 22,50 \\ -13,50 \\ -8,50 \end{pmatrix} \tag{11.1}$$

Onde M representa a *média global* da resposta, T e C os *efeitos principais* (temperatura e catalisador), e TC o *efeito da interação*. Em síntese, os resultados sugerem que 1) a temperatura favoreceu o rendimento, especialmente para o catalisador A, 2) há redução do rendimento quando se substitui o catalisador A pelo B, e c) os maiores rendimentos são obtidos com o catalisador A na temperatura mais alta. Expressando-se esses resultados na função linear múltipla:

$$y = 67,75 + 22,5 * T - 13,5 * C - 8,5 * T * C \tag{11.2}$$

Curiosamente, chega-se aos mesmos resultados se, ao invés de aplicarmos a Equação 10.7, utilizarmos a Equação 10.11, seguido de multiplicação (e não divisão) por outro vetor (*c(1,2,2,2)*:

```
beta <- (solve(t(X) %*% X) %*% t(X) %*% Y) * c(1, 2, 2, 2)
beta
```

```
       [,1]
[1,]  67.75
[2,]  22.50
[3,] -13.50
[4,]  -8.50
```

Sec 11.2 Metodologia de Superfície de Resposta (MSR)

Esta técnica estatística multivariável também é comumente empregada na modelagem de respostas influenciadas por mais de um fator, por vezes associada ao planejamento fatorial, e com vistas à otimização de uma resposta sem a necessidade de se avaliar todas as combinações possíveis. Isso pode ser particularmente útil quando se deseja otimizar um ensaio cuja resposta depende, por exemplo, de preditores escalares, como a faixa de concentração de determinado reagente e as condições de pH.

Exemplificando para uma *superfície de resposta linear* (Neto, Scarminio, e Bruns 2010), num ensaio em que se deseja verificar o melhor rendimento (*rend, %*) de uma reação variando-se 3 níveis de concentração de reagente (*conc, %*) e três velocidades de agitação magnética (*agit, rpm*), pode-se como dantes elaborar a *matriz de contrastes* a partir das informações da tabela que segue:

```
# Dados para metodologia de superfície de resposta

conc <- c(45, 55, 45, 55, 50, 50, 50)
agit <- c(90, 90, 110, 110, 100, 100, 100)
x1 <- c(-1, 1, -1, 1, 0, 0, 0)
x2 <- c(-1, -1, 1, 1, 0, 0, 0)
rendim <- c(69, 59, 78, 67, 68, 66, 69)

tab.msr <- data.frame(conc, agit, x1, x2, rendim)
```

```
knitr::kable(tab.msr, caption = "Dados de experimento de
             metodologia de superfície de resposta linear
             (Neto e cols, 2010).", "pipe") # tabela
```

Tabela 11.2: Dados de experimento de metodologia de superfície de resposta linear (Neto e cols, 2010).

conc	agit	x1	x2	rendim
45	90	-1	-1	69
55	90	1	-1	59
45	110	-1	1	78
55	110	1	1	67
50	100	0	0	68
50	100	0	0	66
50	100	0	0	69

```
# Criação da matriz de coeficientes de contraste
X <- matrix(c(1, 1, 1, 1, 1, 1, 1, -1, 1, -1, 1, 0, 0, 0, -1, -1, 1,
              1, 0, 0, 0), nrow = 7, byrow = FALSE)

# Criação da matriz de rendimento
Y <- as.matrix(rendim)

# Determinação dos coeficientes beta:
beta <- solve(t(X) %*% X) %*% t(X) %*% Y
beta
```

```
      [,1]
[1,] 68.00
[2,] -5.25
[3,]  4.25
```

Dessa forma, a função linear que expressa a superfície de resposta será:

$$y = 68,00 - 5,25 * conc + 4,25 * agit \tag{11.3}$$

11.2.1 Superfície Quadrática de Resposta

Por vezes o modelo linear pode não adequar-se ao planejamento experimental, o que pode ser verificado por uma *Análise de Variância (ANAVA)* do experimento. Nesses casos pode-se aplicar uma *metodologia de superfície quadrática*, e que pode ser expressa como visto na Equação 10.37. Nesses casos é usual uma construção denominada *planejamento em estrela* que acrescenta ao planejamento inicial um idêntico rotacionado 45^o, e cujos pontos novos estão distantes $\sqrt{2}$ unidades codificadas do ponto central. O exemplo abaixo pretende exemplificar essa metodologia.

```
# Dados para superfície quadrática de resposta

conc <- c(30, 40, 30, 40, 35, 35, 35, 28, 35, 42, 35)
agit <- c(115, 115, 135, 135, 125, 125, 125, 125, 139, 125, 119)
x1 <- c(-1, 1, -1, 1, 0, 0, 0, -sqrt(2), 0, sqrt(2), 0)
x2 <- c(-1, -1, 1, 1, 0, 0, 0, 0, sqrt(2), 0, -sqrt(2))
rendim <- c(86, 85, 78, 84, 90, 88, 89, 81, 80, 86, 87)

tab.msr2 <- data.frame(conc, agit, x1, x2, rendim)
knitr::kable(tab.msr2, caption = "Dados de experimento de metodologia de
             superfície quadrática de resposta (Neto e cols, 2010).", "pipe")
```

11

Tabela 11.3: Dados de experimento de metodologia de superfície quadrática de resposta (Neto e cols, 2010).

conc	agit	x1	x2	rendim
30	115	-1.000000	-1.000000	86
40	115	1.000000	-1.000000	85
30	135	-1.000000	1.000000	78
40	135	1.000000	1.000000	84
35	125	0.000000	0.000000	90
35	125	0.000000	0.000000	88
35	125	0.000000	0.000000	89
28	125	-1.414214	0.000000	81
35	139	0.000000	1.414214	80
42	125	1.414214	0.000000	86
35	119	0.000000	-1.414214	87

```
# tabela
```

Dessa vez a *matriz de coeficentes de contrastes* expande-se para seis colunas em função dos termos x_1^2, x_2^2, e $x_1 x_2$, tornando-se:

$$\begin{pmatrix} 1 & -1 & -1 & 1 & 1 & 1 \\ 1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -\sqrt{(2)} & 0 & 2 & 0 & 0 \\ 1 & 0 & \sqrt{(2)} & 0 & 2 & 0 \\ 1 & \sqrt{(2)} & 0 & 2 & 0 & 0 \\ 1 & 0 & -\sqrt{(2)} & 0 & 2 & 0 \end{pmatrix}$$

{#eq-msr2)

O procedimento para a supefície quadrática repete a operação matricial realizada com a linear:

```
# Criação da matriz de coeficientes de contraste para a superfície quadrática
X <- matrix(c(rep(1, 11), -1, 1, -1, 1, 0, 0, 0, -sqrt(2),
              0, sqrt(2), 0, -1, -1, 1, 1, 0, 0, 0, 0, sqrt(2),
              0, -sqrt(2), 1, 1, 1, 1, 0, 0, 0, 2, 0, 2, 0, 1, 1,
              1, 1, 0, 0, 0, 0, 2, 0, 2, 1, -1, -1, 1, 0, 0, 0, 0,
              0, 0, 0), nrow = 11, byrow = FALSE)

# Criação da matriz de rendimento
Y <- as.matrix(rendim)
# Determinação dos coeficientes beta:
beta <- solve(t(X) %*% X) %*% t(X) %*% Y
beta
```

```
          [,1]
[1,] 89.000000
[2,]  1.508883
[3,] -2.362437
[4,] -2.812500
[5,] -2.812500
[6,]  1.750000
```

O resultado expressa função quadrática de superfície encontrada, tal como segue:

$$y = 89,00 + 1,51 * x_1 - 2,36 * x_2 - 2,81 * x_1^2 - 2,81 * x_2^2 + 1,75 * x_1 * x_2 \tag{11.4}$$

Pelo resultado acima é possível prever-se as condições otimizadas para o ensaio. Nesse sentido, o R permite, por exemplo, a construção de um gráfico tridimensional que represente a função obtida, e sem a necessidade de pacote adicional: tal como segue:

```
## Superfície quadrática de resposta (MSR)

x <- seq(-1, 1, 0.1) # preditor x
y <- seq(-1, 1, 0.1) # preditor y
mrs <- function(x, y) {
  89.00 + 1.51 * x - 2.36 * y - 2.81 * x^2 - 2.81 * y^2 + 1.75 * x * y
} # função aplicada aos preditores
z <- outer(x, y, mrs) # saída do gráfico 3D (resposta)
res <- persp(x, y, z, xlab = "x1", ylab = "x2", zlab = "z",
             shade = 0.4, theta = 30, phi = 15, ticktype = "detailed")
# plotagem de superfície da função z(x,y)
pontos <- trans3d(x1, x2, rendim, pmat = res) # comando para adição de pontos experimentais
points(pontos, pch = 19, col = 1) # adição dos pontos
```

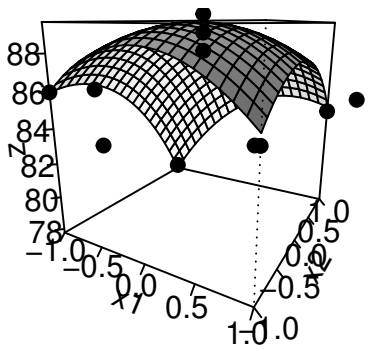


Figura 11.1: Superfície quadrática descrita pela equação de MSR com superposição dos valores experimentais.

Não obstante, existem para o R alguns pacotes para representação tridimensional de dados e funções (rgl, plot3D, scatterplot3d), como também para análise de planejamento fatorial (agricolae,afex,FMC), e de metodologia de superfície de resposta (rsm‘).

11

Sec 12.1 Rotas metabólicas & Balanceamento de Reações

O metabolismo se dá uma intrincada rede de reações químicas catalisadas (ou não) por enzimas diversas, resultando numa rede equilibrada e dinâmica de processos autocatalíticos. Por vezes essa teia de reações pode ser observada no sem número dos chamados *mapas metabólicos* disponíveis, impressos ou na internet. Não obstante, os caminhos metabólicos são classificados para melhor compreensão em subconjuntos de reações pertinentes a determinado grupo de nutrientes ou compostos biológicos, como carboidratos, lipídios e ácidos nucleicos, por ex.

E numa visão mais ampliada desses subconjuntos, uma classificação ulterior resulta por diversas em reações bioquímicas sequenciais e interdependentes, as chamadas *vias ou rotas metabólicas*. Essas vias metabólicas sumarizam processos *catabólicos* ou *anabólicos* envolvendo substratos, produtos, enzimas, cofatores e coenzimas, tal como ilustrado na *glicólise, gliconeogênese, glicogenólise, ciclo do ácido cítrico, cadeia respiratória, via das pentoses,* β-oxidação de ácidos graxos, entre outras.

Dessa forma pode-se considerar uma rota metabólica como uma *combinação linear de reações* catalisadas enzimaticamente (ex: glicólise anaeróbia). Em adição, também pode-se considerar as *reações bioquímicas como equações bioquímicas*, e portanto como um *sistema linear de equações* bioqumímicas com resolução por *álgebra linear*. Assim, pode-se empregar *relações matriciais* para solucionar o balanceamento estequiométrico (massa e carga), obtendo-se a reação líquida final a partir de um conjunto de reações conhecidas. Em síntese, *equações bioquímicas* como *equações matriciais* (Alberty 1991).

Sec 12.2 Operação matricial

Aplicando-se álgebra linear é possível obter-se o balanceamento final de reações sequenciais pela relação que segue, também vista no Capítulo 10:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & ... & a_{1n} \\ a_{21} & a_{22} & a_{23} & ... & a_{2n} \\ a_{31} & a_{32} & a_{33} & ... & a_{3n} \\ ... & ... & ... & ... & a_{jn} \end{pmatrix}, \; b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ ... \\ b_n \end{pmatrix}, \; y = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ ... \\ y_n \end{pmatrix} \tag{12.1}$$

Nesse caso, A representa a *matriz estequiométrica de reações* e compostos, b o *vetor de rota metabólica*, e y o *vetor da reação líquida* balanceado. O vetor de rota indica o número de vezes necessário para cada reação ocorrer, de modo a produzir a reação líquida final. A matriz A é disposta de tal forma a apresentar cada reação em cada coluna, e cada reagente em cada linha, prenchendo-a com os *coeficientes estequiométricos* de reagentes e produtos. Para esses, é necessário apresentar sinal positivo para produtos (são formados) e negativo para reagentes (são consumidos).

A operação matricial para a obtenção da reação líquida envolve apenas um produto cruzado, tal que:

$$A * b = y \tag{12.2}$$

12.2.1 Obtenção do vetor de rota metabólica para a glicólise aeróbia:

Supondo-se conhecer a reação líquida resumida para um conjunto de reações metabólicas, como a glicólise aeróbia (por sua vez agregando glicólise anaeróbia, ciclo do ácido cítrico e fosforilação oxidativa), pode-se facilmente obter-se o vetor de rotas. A partir desse vetor de rotas calculado, é possível obter-se o a reação líquida final do

12

conjunto reações mais detalhado. Na prática, esse detalhamento envolve a participação de coenzimas de oxi-redução, ADP, ATP, e Pi.

Exemplificando, considere as reações abaixo, referentes à uma síntese da glicólise (Alberty 1996):

$$glicose \rightleftarrows 2\,piruvatopiruvato + CoA \rightleftarrows acetil\,CoA + CO_2 acetil\,CoA + 3H_2O \rightleftarrows 2CO_2 + CoA\frac{1}{2}O_2 \rightleftarrows H_2O \tag{12.3}$$

$$acetil\,CoA + 3H_2O \rightleftarrows 2CO_2 + CoA\frac{1}{2}O_2 \rightleftarrows H_2O$$

As reações sequenciais acima possuem como reação líquida resultante:

$$glicose + 6O_2 \rightleftarrows 6CO_2 + 6H_2O \tag{12.4}$$

Para se obter o vetor de rotas então, é necessário elaborar a relação matricial, tal que:

$$A = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 2 & -1 & 0 & 0 \\ 0 & 0 & -3 & 1 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & -1/2 \end{pmatrix} \tag{12.5}$$

$$y = \begin{pmatrix} -1 \\ 0 \\ 6 \\ 0 \\ 0 \\ 6 \\ -6 \end{pmatrix} \tag{12.6}$$

Dessa forma, a solução para o vetor de rotas fica:

$$b = A^{-1} * y \tag{12.7}$$

Ocorre que o sistema linear para o conjunto de equações da via glicolítica é sobrestimado (há mais equações do que incógnitas), não permitindo a solução pela função `solve` de forma direta, como ilustrada no capítulo Capítulo 10. Nesse caso, o vetor b pode ser obtido por solução de mínimos quadrados:

$$b = (A^T * A)^{-1} * A^T * y \tag{12.8}$$

Resolvendo o vetor de rotas no `R`:

```
#  Solução matricial para o vetor de rotas metabólicas

A <- matrix(c(-1, 0, 0, 0, 2, -1, 0, 0, 0, 0, -3, 1, 0, -1, 1, 0, 0, 1,
              -1, 0, 0, 1, 2, 0, 0, 0, 0, -1 / 2), nrow = 7,
            byrow = TRUE) # matriz A de estequiometria de reações
rownames(A) <- list("glicose", "piruvato", "H2O", "CoA", "acetil CoA"
                    , "CO2", "O2") # etiquetas dos reagentes
A # matriz A de reações
```

```
           [,1] [,2] [,3] [,4]
glicose      -1    0    0  0.0
piruvato      2   -1    0  0.0
H2O           0    0   -3  1.0
CoA           0   -1    1  0.0
acetil CoA    0    1   -1  0.0
CO2           0    1    2  0.0
O2            0    0    0 -0.5
```

```
y <- c(-1, 0, 6, 0, 0, 6, -6) # vetor y de rotas

solve(t(A) %*% A) %*% t(A) %*% y
```

```
     [,1]
[1,]    1
[2,]    2
[3,]    2
[4,]   12
```

12.2.2 Obtenção do balanceamento de ATP, ADP, Pi e coenzimas de oxi-redução na glicólise

Com o vetor de rotas obtido (*{1,2,2,12}*), agora é possível aplicá-lo a um conjunto mais extenso de equações da glicólise, para dessa vez solucionar o balanceamento das reações envolvendo coenzimas de oxi-redução, ATP, ADP, e Pi:

$$glicose + 2Pi + 2ADP + 2NAD^+ \rightleftarrows 2\,piruvato + 2ATP + 2NADH + 2H_2O\,, glicólise\,anaeróbia,$$

$$piruvato + CoA + NAD^+ \rightleftarrows acetil - CoA + CO_2 + NADH$$

$$acetil - CoA + 4NAD^+ + ADP + Pi + 2H_2O \rightleftarrows 2CO_2 + 4NADH + ATP + CoA,\, piruvato\,desidrogenase$$

$$NADH + \frac{1}{2}O_2 + 3Pi + 3ADP \rightleftarrows NAD^+ + 4H_2O + 3ATP\,, oxidação\,fosforilativa \tag{12.9}$$

Vale ressaltar uma simplificação pela substituição de GTP e GDP por ATP e ADP (ciclo de Krebs), uma vez que são interconversíveis no metabolismo. Também se assume para simplificação a interconversão de FAD e NAD na reação líquida (Alberty 1996) :

$$FADH + NAD^+ \rightleftarrows FAD + NADH \tag{12.10}$$

De posse das reações presentes na Equação 12.9 e do vetor de rotas obtido anteriormente, pode-se construir a nova matriz estequiométrica e aplicar a solução de mínimos quadrados (Equação 12.8) para se obter a reação líquida da glicólise. E para isso basta aplicar a Equação 12.2) de produto cruzado:

```
A <- matrix(c(-1, 0, 0, 0, -2, 0, -1, -3, -2, 0, -1, -3, -2, -1,
              -4, 1, 2, -1, 0, 0, 2, 0, 1, 3, 2, 1, 4, -1, 2, 0,
              -2, 4, 0, -1, 1, 0, 0, 1, -1, 0, 0, 1, 2, 0, 0, 0,
              0, -1 / 2), nrow = 12, byrow = TRUE)
# matriz A de estequiometria de reações
rownames(A) <- list("glicose", "Pi", "ADP", "NAD", "piruvato", "ATP",
                    "NADH", "H2O", "CoA", "acetil CoA", "CO2", "O2")
# etiquetas dos reagentes
A # matriz A de reações
```

```
           [,1] [,2] [,3] [,4]
glicose      -1    0    0  0.0
Pi           -2    0   -1 -3.0
ADP          -2    0   -1 -3.0
```

```
NAD          -2   -1   -4  1.0
piruvato      2   -1    0  0.0
ATP           2    0    1  3.0
NADH          2    1    4 -1.0
H2O           2    0   -2  4.0
CoA           0   -1    1  0.0
acetil CoA    0    1   -1  0.0
CO2           0    1    2  0.0
O2            0    0    0 -0.5
```

```
y <- c(1, 2, 2, 12) # vetor y de rotas

A %*% y
```

```
           [,1]
glicose      -1
Pi          -40
ADP         -40
NAD           0
piruvato      0
ATP          40
NADH          0
H2O          46
CoA           0
acetil CoA    0
CO2           6
O2           -6
```

```
# solve(t(A)%*%A)%*%t(A)%*%y
```

A partir desse resultado pode-se montar a reação líquida final da via glicolítica como:

$$glicose + 6O_2 + 40ADP + 40Pi \rightleftarrows 6CO_2 + 40ATP \tag{12.11}$$

O exemplo acima ilustra o emprego de álgebra matricial para a solução de problemas de balanceamento de reações bioquímicas. Por sua natureza trata-se de método abrangente e de natureza algorítmica (portanto, programável), embora não seja o único. Outras propostas de solução para o balanceamento e conservação de massa e carga englobam a *inspeção direta* por triagem e erro a partir de regras mnemônicas, o balanceamento por *método de meia-equação*, e o *método de número de oxidação*, não discutidos aqui.

Podemos considerar o metabolismo celular como uma rede de vias metabólicas retroalimentadas homeostaticamente. Se observarmos com ampliação qualquer uma das reações bioquímicas envolvidas nessa teia complexo, poderemos identificar uma *caixa preta* comum a todas:

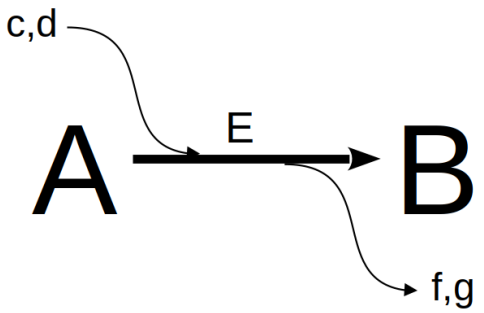


Figura 13.1: Uma representação da caixa preta de reações enzimáticas. E = enzima c,d = cofatores, coenzimas, modificadores; f,g = compostos secundários resultantes da catálise .

Ainda que a representação acima possa envolver reações em equilíbrio (duplas setas), ou mesmo a ausência do catalisador, a imagem generaliza reações bioquímicas individuais partícipes de qualquer *mapa metabólico*. Se desejarmos agora avaliar o consumo do composto A (reagente, substrato) e resultante formação do composto B (produto), ou seja, os teores dos compostos num intervalo de tempo, podemos considerar, como visto no Capítulo 5), a reação acima como de 1a. ordem em ambos, reagente e produto:

$$A \overset{k}{\rightarrow} B \tag{13.1}$$

Dessa forma, pode-se considerar a velocidade de reação como a variação de A ou B no tempo como:

$$v = \frac{dy}{dx} = -\frac{dA}{dt} = \frac{dB}{dt} \tag{13.2}$$

Ou seja, quando há consumo, a taxa de variação apresenta-se negativa, e quando há formação, positiva. Separando as duas taxas:

$$\frac{dA}{dt} = -k * A; \frac{db}{dt} = k * B \tag{13.3}$$

E dessa forma, para se calcular a variação em cada composto ao longo de um intervalo de tempo:

$$dA = -k * A * dt; dB = k * A * dt \tag{13.4}$$

Observe que estamos diante de um *sistema de equações diferenciais*, nominalmente *ordinárias*, já que as alterações no tempos ocorrem um função de um único parâmetro (concentração das espécies). Caso constituisse de um sistema dependente em mais de um parâmetro, estaríamos tratando de equações diferenciais *parciais* (comuns nas relações termodinâmicas que envolvem modificações com variação de volume, temperatura, e pressão).

13

Algumas equações diferenciais podem ser analiticamente resolvidas, como as que envolvem o crescimento exponencial bacteriano:

$$\frac{dN}{dt} = -k * N;\ N(t) = N_0 * e^{-kt}; (N = N_0\, em\, t = 0) \tag{13.5}$$

Sec 13.1 Solução numérica para sistema de equações diferenciais

Por outro lado, quando uma equação ou sistema de equações diferenciais possui certa complexidade para a solução analítica, busca-se a solução numérica. Ainda que existam diversas bibliotecas para a solução de equações diferenciais pelo `R` (`deSolve`,`pracma`, `lsoda`), alguns sistemas simples podem ser resolvidos com os pacotes básicos de instalação:

O procedimento mais simples emprega o *método de Euler*. A ideia básica do método consiste em integrar uma função diferencial de variação infinitesimal na variável independente (no caso, o tempo), para uma relação real, e a partir de valores iniciais fornecidos. Simplificando, o valor da função corresponderá ao acréscimo do incremento *dy* para cada intervalo *dx*, a partir da relação de cada reação envolvida na transformação dos compostos. Exemplificando para as reações presentes na Equação 13.4:

```
# Solução de equações diferencias para conversão A-->B

k <- 0.5 # constante cinética de catálise
dt <- .005
tmax <- 3 # intervalo de tempo & tempo máximo
t <- seq(0, tmax, dt) # vetor de tempo
n <- tmax / dt + 1 # no. de pontos da simulação (necessário o acréscimo
# de 1 para que vetores fiquem de mesmo tamanho)
x <- matrix(rep(0, 2 * n), nrow = 2, ncol = n) # construção da matriz de
# uma linha pra cada composto, e uma coluna pra cada tempo dt
x[1, 1] <- 1
x[2, 1] <- 0 # valores iniciais de concentração
for (i in 2:n) {
  dA <- -k * x[1, i - 1] * dt # dA
  dB <- k * x[1, i - 1] * dt # dB
  x[1, i] <- x[1, i - 1] + dA # variação em A com acrécimo dA
  x[2, i] <- x[2, i - 1] + dB # variação em B com acréscimo dB
  # laço que acrescenta a cada intervalo dt o valor do novo teor para
  # cada composto
}
plot(t, x[1, ],
  type = "l", lty = 1,
  xlab = "tempo, s", ylab = "[espécie], M", ylim = c(0, 1.025),
  bty = "l"
) # gráfico do composto 1
lines(t, x[2, ], lty = 2, col = 2) # adição do gráfico do composto 2
legend(
  x = 2, 5, y = 1, legend = c("A", "B"), col = c(1, 2),
  cex = 1,  lty = c(1, 2)
)
```

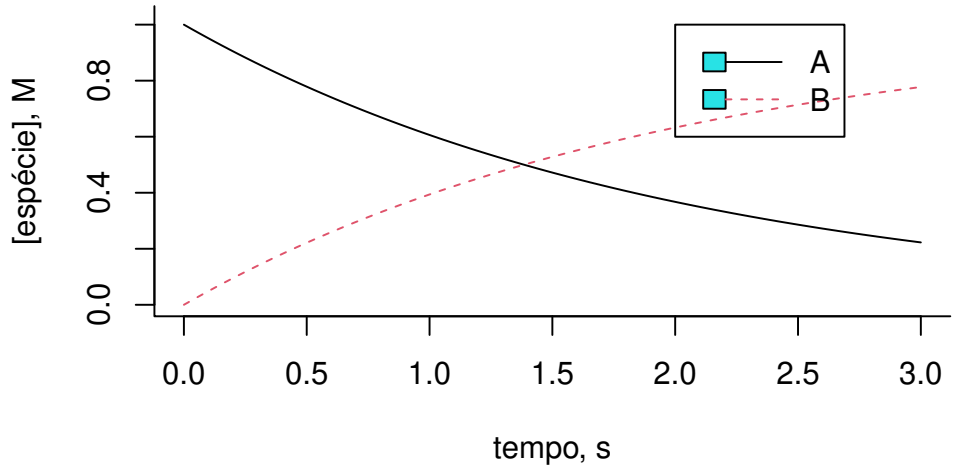


Figura 13.2: Solução de sistema de equações diferenciais por método de Euler para conversão de 1a. ordem da espécie A em B, a uma taxa cinética k.

Experimente variar a constante cinética k, ou os valores iniciais para cada composto, e observe o efeito resultante. As reações metabólicas por diversas vezes apresentam interconversões entre compostos, tal que um substrato da reação também pode configurar-se como produto de catálise da mesma, e com taxas cinéticas de síntese (k) e degradação (km, ou k *minus*) para cada composto, como segue:

$$A \underset{km}{\overset{k}{\rightleftharpoons}} B \tag{13.6}$$

Nesse caso, o sistema de equações diferenciais ficará:

$$dA = -k * A * dt + km * B * dt; dB = k * A * dt - km * B * dt \tag{13.7}$$

Implementando-se o trecho de código no R:

```
k <- 0.5
km <- 0.5 # constantes cinéticas de catálise
dt <- .005
tmax <- 10 # intervalo de tempo & tempo máximo
t <- seq(0, tmax, dt) # define vetor de tempo
n <- tmax / dt + 1 # define no. de pontos
x <- matrix(rep(0, 2 * n), nrow = 2, ncol = n) # constroi matriz de uma
# linha pra cada composto, e uma coluna pra cada tempo dt
x[1, 1] <- 1
x[2, 1] <- 1 # valores iniciais de concetração
for (i in 2:n) {
  dA <- -k * x[1, i - 1] * dt + km * x[2, i - 1] * dt
  dB <- k * x[1, i - 1] * dt - km * x[2, i - 1] * dt
  x[1, i] <- x[1, i - 1] + dA
  x[2, i] <- x[2, i - 1] + dB
  # laço que acrescenta a cada intervalo dt o valor de novo teor para
  # cada composto
}
plot(t, x[1, ],
```

```
  type = "l", lty = 1,
  xlab = "tempo, s", ylab = "[espécie], M", ylim = c(0, 2), bty = "l"
) # gráfico do composto 1
lines(t, x[2, ], lty = 2, col = 2) # adição do gráfico do composto 2
legend(
  x = 2, 5, y = 2, legend = c("A", "B"), col = c(1, 2), cex = 1,
  lty = c(1, 2)
)
```

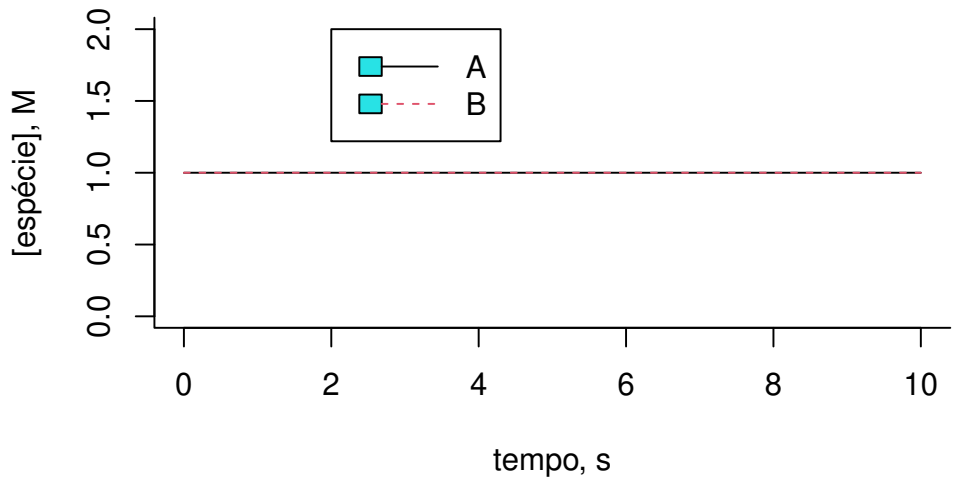


Figura 13.3: Solução numérica para a conversão reversível da espécie A em B. k = km = 0,5; Ao e Bo = 1 (teores iniciais).

Observe que os teores de A e B permanecem constantes ao longo do intervalo. Isso decorre dos valores idênticos das constantes cinéticas para cada reação direta e reversa, bem como dos teores iniciais para cada composto. Ilustrando uma variação desses:

```
# Exemplo de conversão A-->B

k <- 0.5
km <- 0.1 # constantes cinéticas de catálise
dt <- .005
tmax <- 10 # intervalo de tempo & tempo máximo
t <- seq(0, tmax, dt) # define vetor de tempo
n <- tmax / dt + 1 # define no. de pontos
x <- matrix(rep(0, 2 * n), nrow = 2, ncol = n) # constroi matriz de
# uma linha pra cada composto, e uma coluna pra cada tempo dt
x[1, 1] <- 1
x[2, 1] <- 0.2 # valores iniciais de concentração
for (i in 2:n) {
  dA <- -k * x[1, i - 1] * dt + km * x[2, i - 1] * dt
  dB <- k * x[1, i - 1] * dt - km * x[2, i - 1] * dt
  x[1, i] <- x[1, i - 1] + dA
  x[2, i] <- x[2, i - 1] + dB
  # laço que acrescenta a cada intervalo dt o valor de novo teor para
  # cada composto
}
plot(t, x[1, ],
```

```
  type = "l", lty = 1,
  xlab = "tempo, s", ylab = "[espécie], M", ylim = c(0, 2), bty = "l"
) # gráfico do composto 1
lines(t, x[2, ], lty = 2, col = 2) # adição do gráfico do composto 2
legend(x = 2, 5, y = 2, legend = c("A", "B"), col = c(1, 2), cex = 1, lty = c(1, 2))
```

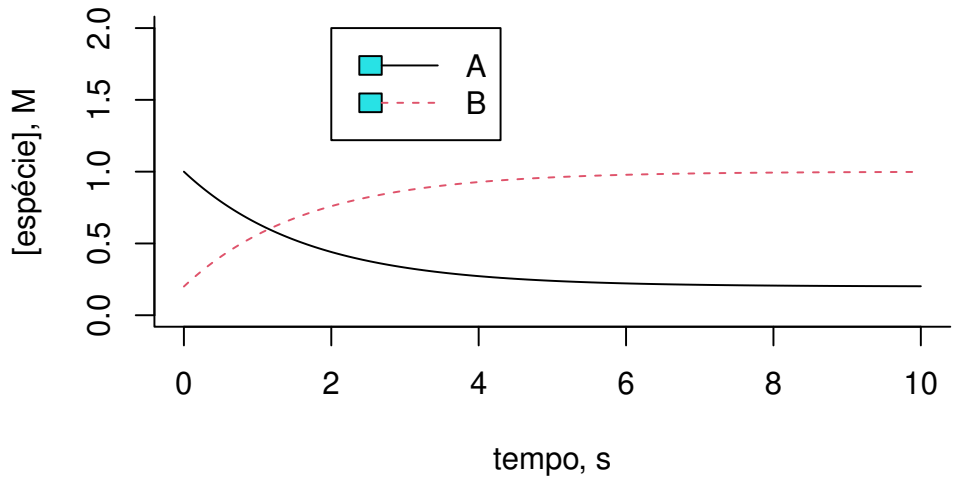


Figura 13.4: Solução numérica para a conversão reversível da espécie A em B. k = 0,5; km = 0,1; Ao = 1; Bo = 0,2 (teores iniciais).

Agora podemos imaginar uma reação um pouco mais complexa, como a ilustrada na Equação 13.8:

$$A \underset{km1}{\overset{k1}{\rightleftharpoons}} B \overset{k2}{\rightarrow} C \tag{13.8}$$

Nesse caso, o sistema de equações diferenciais será:

$$dA = -k1 * A * dt + km1 * B * dt; dB = k1 * A * dt - km1 * B * dt - k2 * B * dt; dC = k2 * B \tag{13.9}$$

Implementando-se o trecho de código:

```
# Solução de Euler para cinética de 3 compostos

k1 <- 0.5
km1 <- 0.1
k2 <- 1 # constantes cinéticas de catálise
dt <- .005
tmax <- 3 # intervalo de tempo & tempo máximo
t <- seq(0, tmax, dt) # define vetor de tempo
n <- tmax / dt + 1 # define no. de pontos
x <- matrix(rep(0, 3 * n), nrow = 3, ncol = n) # constroi matriz de
# uma linha pra cada composto, e uma coluna pra cada tempo dt
x[1, 1] <- 1
x[2, 1] <- 0
x[3, 1] <- 0 # valores iniciais de concentração
for (i in 2:n) {
```

```
  dA <- -k1 * x[1, i - 1] * dt + km1 * x[2, i - 1] * dt
  dB <- k1 * x[1, i - 1] * dt - (km1 + k2) * x[2, i - 1] * dt
  dC <- k2 * x[2, i - 1] * dt
  x[1, i] <- x[1, i - 1] + dA
  x[2, i] <- x[2, i - 1] + dB
  x[3, i] <- x[3, i - 1] + dC # laço que acrescenta a cada intervalo dt
  # o valor de novo teor para cada composto
}
plot(t, x[1, ],
  type = "l", lty = 1,
  xlab = "tempo, s", ylab = "[espécie], M", ylim = c(0, 1.025), bty = "l"
) # gráfico do composto 1
lines(t, x[2, ], lty = 2, col = 2) # adição do gráfico do composto 2
lines(t, x[3, ], lty = 3, col = 3) # adição do gráfico do composto 3
legend(x = 2, 5, y = 1, legend = c("A", "B", "C"), col = c(1, 2, 3),
       cex = 1, lty = c(1, 2, 3))
```

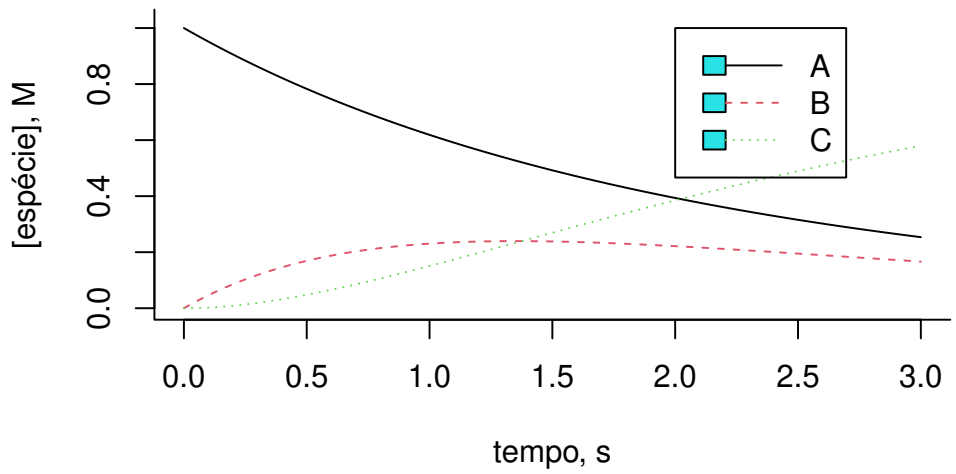


Figura 13.5: Solução de Euler para uma cinética de 3 compostos. k1 = 0,5; k2 = 1; km1 = 0,1. Teores iniciais: Ao = 1; Bo = 0; Co = 0.

Observe que a Equação 13.9 acima reflete uma catálise de Michaelis-Mentem, embora considerando o teor da enzima E como constante e, portanto, independente da reação (ordem zero). E observe também que o gráfico da Figura 13.5 traduz, de certa forma, a condição Briggs-Haldane do estado estacionário tratada no Capítulo 5 sobre *Enzimas*. Note que a variação de B, refletida nessa condição como o complexo ES, mantém-se relativamente constante por determinado intervalo de tempo, sendo produzida pela colisão com a enzima E, e desconstruida tanto por sua conversão a $E + P$ (no caso, C), como por sua reversão a $E + S$ (no caso, A).

Para uma reação um pouco mais complexa:

$$A \underset{km1}{\overset{k1}{\rightleftharpoons}} B \underset{km2}{\overset{k2}{\rightleftharpoons}} C \tag{13.10}$$

O que sugere o seguinte trecho de código no R:

```
# Solução de Euler para cinética reversível de 3 compostos
```

```
# constantes da reação direta
k1 <- 3
km1 <- 1
k2 <- 4
km2 <- 0.7
dt <- .005
tmax <- 10
t <- seq(0, tmax, dt)
n <- tmax / dt + 1
x <- matrix(rep(0, 3 * n), nrow = 3, ncol = n)
x[1, 1] <- 1
x[2, 1] <- 0
x[3, 1] <- 0
for (i in 2:n) {
  dA <- -k1 * x[1, i - 1] * dt + km1 * x[2, i - 1] * dt
  dB <- k1 * x[1, i - 1] * dt - (km1 + k2) * x[2, i - 1] * dt +
    km2 * x[3, i - 1] * dt
  dC <- k2 * x[2, i - 1] * dt - km2 * x[3, i - 1] * dt
  x[1, i] <- x[1, i - 1] + dA
  x[2, i] <- x[2, i - 1] + dB
  x[3, i] <- x[3, i - 1] + dC
}
plot(t, x[1, ],
  type = "l", lty = 1,
  xlab = "tempo, s", ylab = "[espécie], mol/L",
  ylim = c(0, 1.025), bty = "l"
)
lines(t, x[2, ], col = 2, lty = 2)
lines(t, x[3, ], col = 3, lty = 3)
legend(x = 8, y = 0.6, legend = c("A", "B", "C"), col = c(1, 2, 3),
       cex = 1, lty = c(1, 2, 3))
```

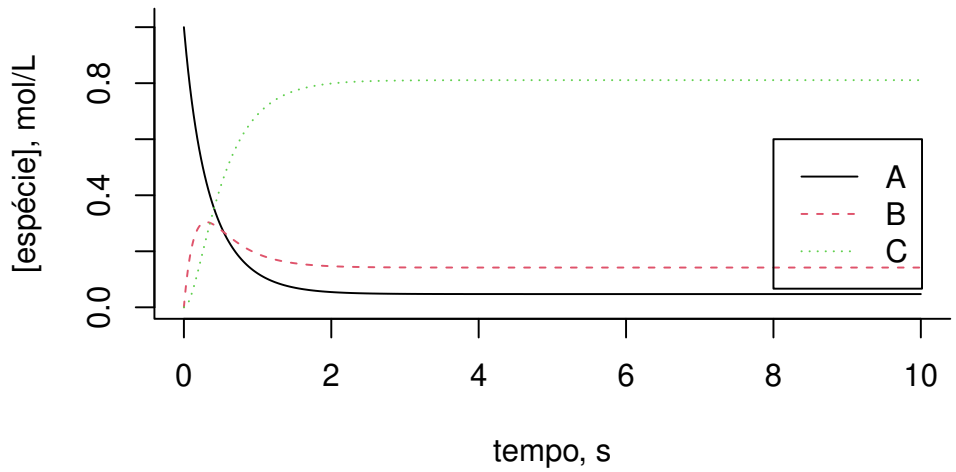


Figura 13.6: Solução de Euler para uma cinética reversível de 3 compostos. k1 = 1; km1 = 3; k2 = 5; km1 = 0,1. Teores iniciais: Ao = 1; Bo = 0; Co = 0.

Agora, suponha uma cadeia mais complexa de reações bioquímicas, com moduladores alostéricos negativos

13

(inibição, ki) e positivos (ativação, ka) para determinadas enzimas. O exemplo abaixo ilustra essa situação:

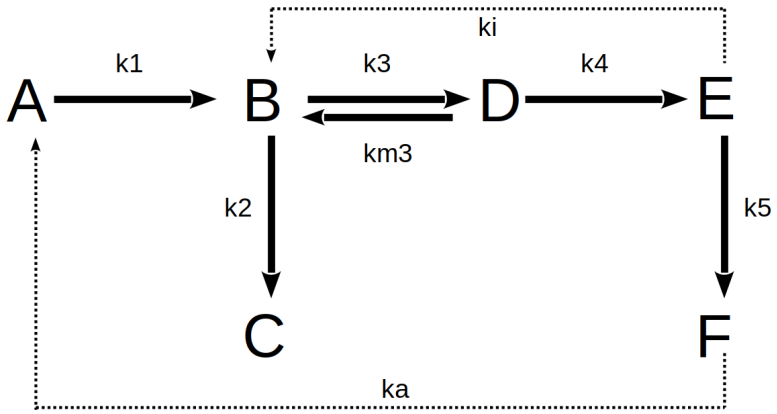


Figura 13.7: Um exemplo de reações bioquímicas em rota metabólica fictícia. As setas pontilhadas juntamente aos valores de ki e ka representam modulações alostéricas com respectivas constantes de inibição e ativação enzimáticas.

Assim, o conjunto de reações da rede metabólica acima pode ser equacionado como:

$$dA = -k1 * A * dt + ka * F * dt; dB = k1 * A * dt + km3 * D * dt - k3 * B * dt - k2 * B * dt - ki * E * dt; dC = k2 * B * dt; \tag{13.11}$$

$$dD = k3 * B * dt - km3 * D * dt - k4 * D * dt; dE = k4 * D * dt - k5 * E * dt; dF = k5 * E * dt - ka * F * dt \tag{13.12}$$

Pode-se elaborar o trecho de código que segue para a solução numérica de Euler que envolve as equações diferenciais elencadas acima como:

```
# Solução para rota metabólica com inibição e ativação alostéricas

# Constantes cinéticas e alostéricas
k1 <- 2
k2 <- 0.5
k3 <- 0.7
km3 <- 0.3
k4 <- 5
k5 <- 1
ki <- 0.3 # constante de inibição
ka <- 0.2 # constante de ativação
dt <- .005
tmax <- 10
t <- seq(0, tmax, dt)
n <- tmax / dt + 1
x <- matrix(rep(0, 6 * n), nrow = 6, ncol = n)
# Valores iniciais dos compostos
x[1, 1] <- 1
x[2, 1] <- 0
x[3, 1] <- 0
x[4, 1] <- 1
x[5, 1] <- 0
x[6, 1] <- 0
for (i in 2:n) {
  # sistema de equações inserido na matriz dos intervalos
  dA <- -k1 * x[1, i - 1] * dt + ka * x[6, i - 1] * dt
  dB <- k1 * x[1, i - 1] * dt + km3 * x[4, i - 1] * dt - k3 *
```

```
    x[2, i - 1] * dt - k2 * x[2, i - 1] * dt - ki * x[1, i - 1] * dt
  dC <- k2 * x[2, i - 1] * dt
  dD <- k3 * x[2, i - 1] * dt - km3 * x[4, i - 1] * dt - k4 *
    x[4, i - 1] * dt
  dE <- k4 * x[4, i - 1] * dt - k5 * x[5, i - 1] * dt
  dF <- k5 * x[5, i - 1] * dt - ka * x[6, i - 1] * dt
  # Adição dy aos valores de y
  x[1, i] <- x[1, i - 1] + dA
  x[2, i] <- x[2, i - 1] + dB
  x[3, i] <- x[3, i - 1] + dC
  x[4, i] <- x[4, i - 1] + dD
  x[5, i] <- x[5, i - 1] + dE
  x[6, i] <- x[6, i - 1] + dF
}
# Elaboração dos gráficos cinéticos
plot(t, x[1, ],
  type = "l", lty = 1,
  xlab = "tempo,s", ylab = "[espécie], mol/L",
  ylim = c(0, 1.025), bty = "l"
)
lines(t, x[2, ], col = 2, lty = 2)
lines(t, x[3, ], col = 3, lty = 3)
lines(t, x[4, ], col = 4, lty = 4)
lines(t, x[5, ], col = 5, lty = 5)
lines(t, x[6, ], col = 6, lty = 6)
legend(x = 6.5, y = 0.65, legend = c("A", "B", "C", "D", "E", "F"),
       col = c(1, 2, 3, 4, 5, 6), cex = 1, lty = c(1, 2, 3, 4, 5, 6))
```

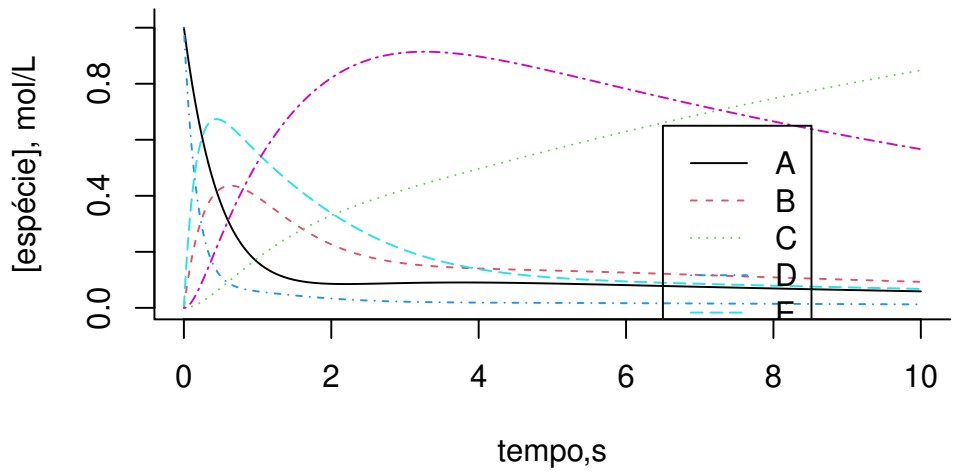


Figura 13.8: Solução para uma via metabólica fictícia apresentando inibição (ki) e ativação (ka) alostéricas. Taxas cinéticas: k1 = 2, k2 = 0,5, k3 = 0,7, km3 = 0,3, k4 = 5, k5 = 1, ki = 0,3, ka = 0.2. Valores iniciais dos compostos: A=1; B, C, D, E, F = 0

Altere as constantes cinéticas e/ou alostéricas do sistema acima e observe o efeito em cada um dos compostos. De modo geral, a solução de Euler aplicada a sistemas de complexidade crescente, como uma rede metabólica, pode apresentar desvios centrados na seleção do valor de *dt*, ou mesmo produzir valores inconsistentes. Para contornar essa situação utiliza-se outros algoritmos, tais como de *Runge-Kutta de 2a., 3a, ou 4a. ordem*, presentes nos pacotes

do R, ou ainda por *análise de sistemas*.

Sec 13.2 Algumas reações do metabolismo da glicose

Para uma aplicação do método de *Runge-Kutta de 4a. ordem* é necessário instalar o pacote `deSolve` ou similar para a solução de sistema de equações diferenciais ordinárias de 1a. ordem ou diferenciais parciais. A biblioteca agrega funções que permitem um código mais enxuto e simples para a solução do sistema. Ilustrando sua aplicação, seguem algumas das muitas relações simples da rede metabólica que envolve a *glicólise, gliconeogênese, e via das pentoses* nas células:

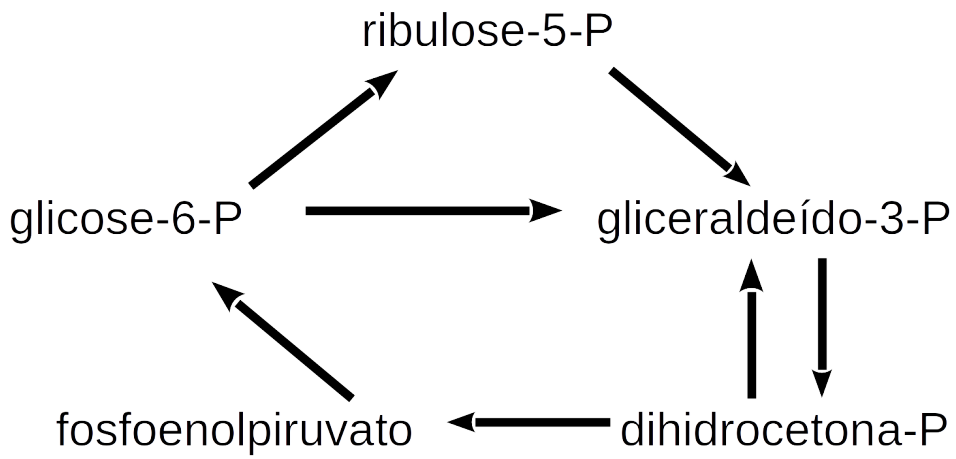


Figura 13.9: Algumas relações metabólicas envolvidas na glicólise, gliconeogênese e vias das pentoses.

Pode-se atribuir a essas relações as seguintes reações do metabolismo:

$$G6P \overset{k1}{\rightarrow} R5PG3P + DHCP \overset{k2}{\rightarrow} PEP2G3P \overset{k3}{\rightarrow} 6GPR5P \overset{k4}{\rightarrow} G3P2PEP \overset{k5}{\rightarrow} G6P \tag{13.13}$$

O trecho de código para a solução por Runge-Kutta pode ser o exemplificado a seguir, com resultados em dois gráficos; inicialmente com as curvas isoladas, e depois reunidas.

```
library(deSolve)

# Solução para cinética de conversão em algumas vias metabólicas

# Parâmetros das reações
k1 <- 0.1
k2 <- 0.5
k3 <- 0.05
k4 <- 0.5
k5 <- 0.2
parms <- c(k1, k2, k3, k4, k5)

# Valores iniciais para cada composto
G6P0 <- 1
R5P0 <- 0
G3P0 <- 0.3
DHCP0 <- 0.1
```

```
PEP0 <- 0
# Intervalo de tempo
tmin <- 0
tmax <- 20
dt <- 0.01
tempo <- seq(tmin, tmax, dt)
# Função para as derivadas das espécies no tempo
eq.dif <- function(tempo, x, parms) {
  # especificação dos compostos
  G6P <- x[1]
  R5P <- x[2]
  G3P <- x[3]
  DHCP <- x[4]
  PEP <- x[5]
  # equações diferenciais
  dG6P <- -k1 * G6P + k3 * G3P^2 + k5 * PEP^2
  dR5P <- k1 * G6P - k4 * R5P
  dG3P <- -k2 * G3P * DHCP - k3 * G3P^2 + k4 * R5P
  dDHCP <- -k2 * G3P * DHCP
  dPEP <- k2 * G3P * DHCP - k5 * PEP^2
  list(c(dG6P, dR5P, dG3P, dDHCP, dPEP)) # incrementos das espécies
}
# Rotina de lsoda pra solução diferencial ordinária

out <- lsoda(c(G6P0, R5P0, G3P0, DHCP0, PEP0), tempo, eq.dif, parms,
  rtol = 1e-4, atol = 1e-6
)
# Saída do resultado em vetores pra cada quantidade (tempo e espécies)
t <- out[, 1]
G6P <- out[, 2]
R5P <- out[, 3]
G3P <- out[, 4]
DHCP <- out[, 5]
PEP <- out[, 6]
# Elaboração de gráficos verticais
par(mfrow = c(1, 5))
plot(t, G6P, type = "l")
plot(t, R5P, type = "l")
plot(t, G3P, type = "l")
plot(t, DHCP, type = "l")
plot(t, PEP, type = "l")
```

13

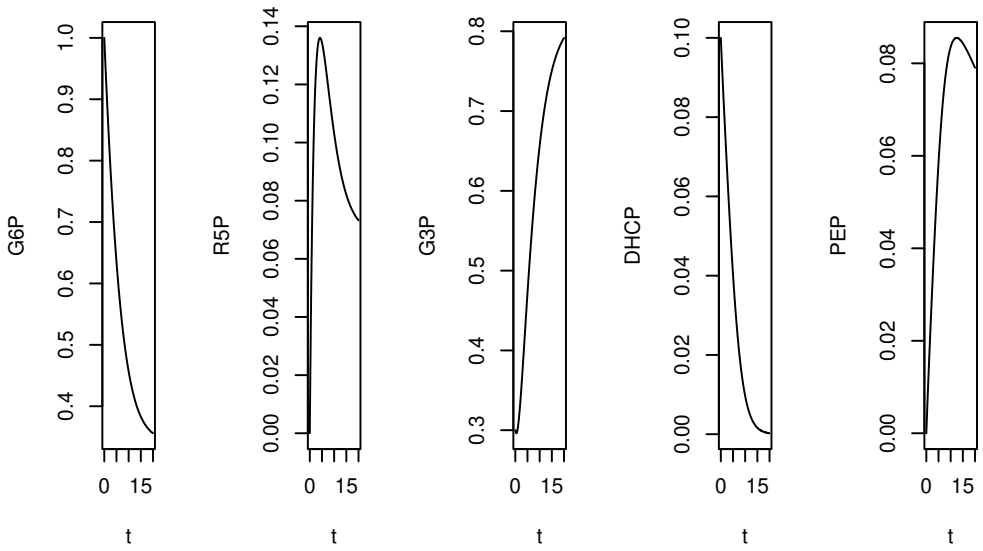


Figura 13.10: Cinética de conversões para uma rede metabólica simples envolvendo algumas reações da glicólise, gliconeogênese e via das pentoses. Valores das constantes cinéticas: k1 = 0,1; k2 = 0,5; k3 = 0,05; k4 = 0,5; k5 = 0,2. Valores iniciais dos compostos: G6P = 1; para os demais, 0.

```
# Elaboração de gráfico com todas as espécies
par(mfrow = c(1, 1))
plot(t, G6P, type = "l", col = 1, lty = 1, ylab = "[espécie]",
     ylim = c(0, 1))
lines(t, R5P, type = "l", col = 2, lty = 2)
lines(t, G3P, type = "l", col = 3, lty = 3)
lines(t, DHCP, type = "l", col = 4, lty = 4)
lines(t, PEP, type = "l", col = 5, lty = 5)
legend(x = 10, y = 1, legend = c("G6P", "R5P", "G3P", "DHCP", "PEP"),
       col = c(1, 2, 3, 4, 5), cex = 1, lty = c(1, 2, 3, 4, 5))
```

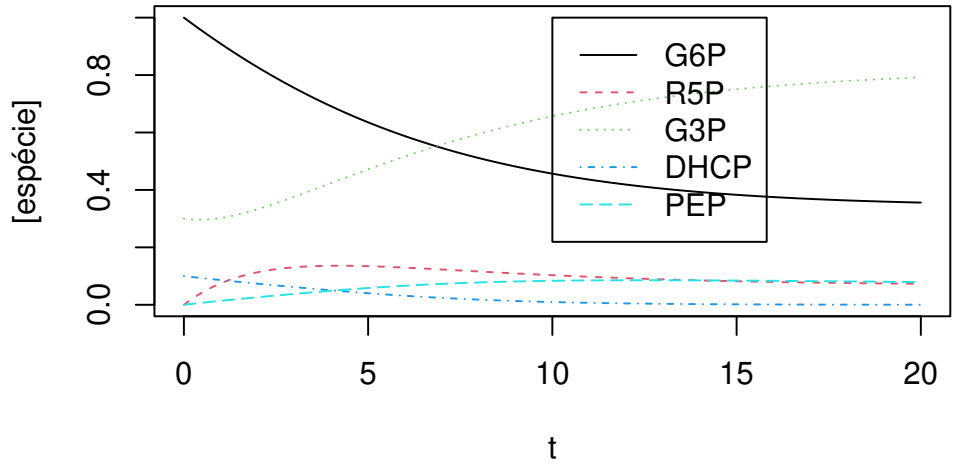


Figura 13.11: Cinética de conversões para uma rede metabólica simples envolvendo algumas reações da glicólise, gliconeogênese e via das pentoses. Valores das constantes cinéticas: k1 = 0,1; k2 = 0,5; k3 = 0,05; k4 = 0,5; k5 = 0,2. Valores iniciais dos compostos: G6P = 1; para os demais, 0.

Perceba que, pelas quantidades oferecidas à simulação, ou seja, constantes cinéticas e valores iniciais, G3P e PEP registram um intervalo significativo em elevação, também coincidente por sua presença em várias das relações da Equação 13.13.

Sec 13.3 Cinética do metabolismo de 6-mercaptopurina

Elevando um pouco a complexidade de redes metabólicas, pode-se exemplificar o metabolismo celular da 6-mercaptopurina (6-MP) em função do teor de ATP celular (Lavrova et al. 2017). Como antagonista da purina, o fármaco é empregado na quimioterapia para o tratamento de leucemia linfocítica, interrompendo o crescimento celular, embora produzindo efeitos colaterais citotóxicos advindos de reações com o grupo tiol.

A Figura 13.12 representa um esquema simplificado do metabolismo de 6-MP. As concentrações das espécies e constantes cinéticas originam-se dos valores de concentração em μmol/mL, e tempo em dias.

Para as 10 reações referentes às ODEs que compõem a representação do metabolismo de 6-MP (Lavrova et al. 2017), o trecho de código abaixo implementa a solução por Runge-Kutta de 4a. ordem pela função `lsoda`:

```
#  Degradação de 6-mercaptopurina e solução de Runge-Kutta

library(deSolve)
# Parâmetros
k0 <- 5
k1 <- 10
k2 <- 10
k3 <- 5
k4 <- 1e-5
k7 <- 0.01
k8 <- 0.5
km7 <- 1
km1 <- 0.01
km2 <- 4
km3 <- 0.01
km4 <- 0.1
```

13

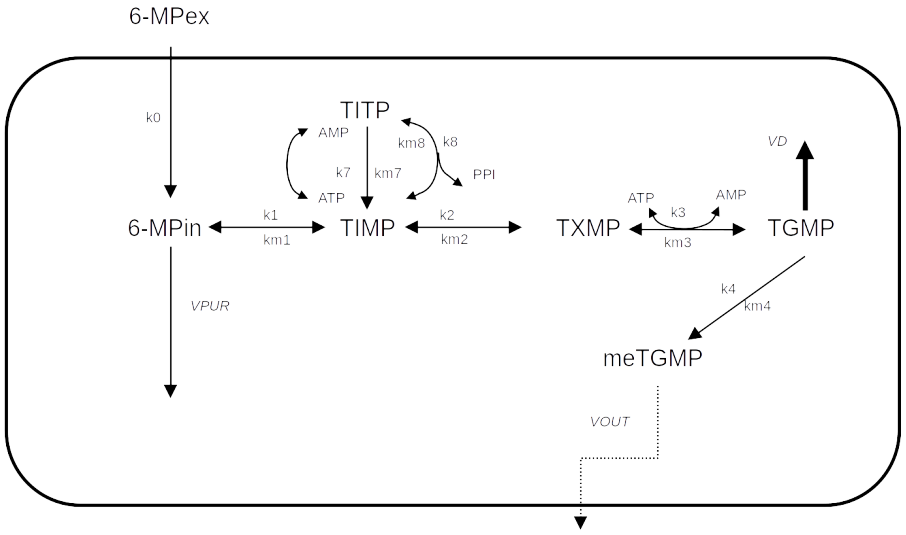


Figura 13.12: Metabolismo esquemático simplificado para o metabolismo de 6-mercaptopurina (adaptado de Lavrova et al., 2017).

```
km8 <- 0.01
VPUR <- 0.01
VD <- 0.9
VOUT <- 1e-4
# Lista de parâmetros
parms <- c(k0, k1, k2, k3, k4, k7, k8, km7, km1, km2, km3, km4, km8,
           VPUR, VD, VOUT)
# especificação dos compostos
MPex <- x[1]
MPin <- x[2]
TIMP <- x[3]
TXMP <- x[4]
TGMP <- x[5]
meTGMP <- x[6]
TITP <- x[7]
ATP <- x[8]
AMP <- x[9]
PP <- x[10]
# Concentrações iniciais das espécies
reag0 <- c(MPex0 = 0.68, MPin0 = 0, TIMP0 = 0, TXMP0 = 0, TGMP0 = 0,
           meTGMP0 = 0, TITP0 = 0, ATP0 = 0.2, AMP0 = 0, PP0 = 0)

# Definição do intervalo de tempo
tmin <- 0
tmax <- 2
dt <- 0.01
tempo <- seq(tmin, tmax, dt)
# Função para as derivadas de cada espécie
eq.dif <- function(tempo, x, parms) {
  # Definição de parâmetros
  MPex <- x[1]
  MPin <- x[2]
  TIMP <- x[3]
  TXMP <- x[4]
  TGMP <- x[5]
  meTGMP <- x[6]
```

```
  TITP <- x[7]
  ATP <- x[8]
  AMP <- x[9]
  PP <- x[10]
  # Equações diferenciais
  dMPex <- -k0 * MPex
  dMPin <- -(VPUR + k1) * MPin + k0 * MPex + km1 * TIMP
  dTIMP <- k1 * MPin + km8 * TITP - (k2 + k7 * ATP + km1 + k8 * PP) *
    TIMP + km2 * TXMP + km7 * TITP * AMP
  dTXMP <- k2 * TIMP - k3 * TXMP * ATP - km2 * TXMP + km3 * TGMP *
    AMP * PP
  dTGMP <- k3 * TXMP * ATP - (k4 + VD) * TGMP - km3 * TGMP * AMP *
    PP + km4 * meTGMP
  dmeTGMP <- k4 * TGMP - VOUT * meTGMP - km4 * meTGMP
  dTITP <- k8 * TIMP * PP - km8 * TITP + k7 * TIMP * ATP - km7 *
    TITP * AMP
  dATP <- -k7 * TIMP * ATP + km3 * TGMP * AMP * PP - k3 * TXMP *
    ATP + km7 * TITP * AMP
  dAMP <- -km3 * TGMP * AMP * PP + k3 * TXMP * ATP + k7 * TIMP *
    ATP - km7 * TITP * AMP
  dPP <- -k8 * TIMP * PP + km8 * TITP - km3 * TGMP * AMP * PP + k3 *
    TXMP * ATP
  list(c(dMPex, dMPin, dTIMP, dTXMP, dTGMP, dmeTGMP, dTITP, dATP,
         dAMP, dPP)) # lista de valores diferenciais para cada espécie
}
# Rotina de lsoda pra solução equações diferenc. ordinárias
sol.eq <- lsoda(reag0, tempo, eq.dif, parms,
  rtol = 1e-4, atol = 1e-6
)
# Isolamento das colunas de resultdos
t <- sol.eq[, 1]
MPex <- sol.eq[, 2]
MPin <- sol.eq[, 3]
TIMP <- sol.eq[, 4]
TXMP <- sol.eq[, 5]
TGMP <- sol.eq[, 6]
meTGMP <- sol.eq[, 7]
TITP <- sol.eq[, 8]
ATP <- sol.eq[, 9]
AMP <- sol.eq[, 10]
PP <- sol.eq[, 11]

# Elaboração do gráfico
plot(t, MPex, type = "l", xlab = "tempo, dias",
     ylab = "[espécie], umol/L")
lines(t, MPin, type = "l", col = 2, lty = 2)
lines(t, TIMP, type = "l", col = 3, lty = 3)
lines(t, TXMP, type = "l", col = 4, lty = 4)
lines(t, TGMP, type = "l", col = 5, lty = 5)
lines(t, meTGMP, type = "l", col = 6, lty = 6)
lines(t, TITP, type = "l", col = 7, lty = 7)
lines(t, ATP, type = "l", col = 8, lty = 8)
lines(t, AMP, type = "l", col = 9, lty = 9)
lines(t, PP, type = "l", col = 10, lty = 10)
legend(x = 1.5, y = 0.65, legend = c("MPex", "MPin", "TIMP", "TXMP",
                                     "TGMP", "meTGMP", "TITP", "ATP",
                                     "AMP", "PP"), col = c(1, 2, 3, 4,
                                                           5, 6, 7, 8,
                                                           9, 10),
```

13

```
        cex = 0.7, lty = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10))
```

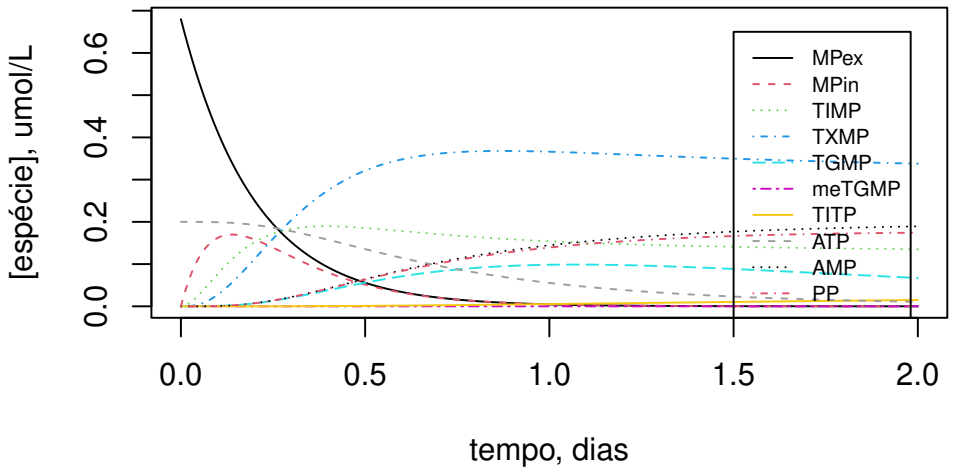


Figura 13.13: Dependência da dinâmica da rede metabólica de degradação de 6-mercaptopurina em função do teor inicial de ATP a 0,2 umol/L. Os valores iniciais e parâmetros são descritos no trecho de código.

Como a dinâmica de variação dos compostos é dependente do teor inicial de ATP celular, experimente variar esse valor inicial (ex: ATP0=2):

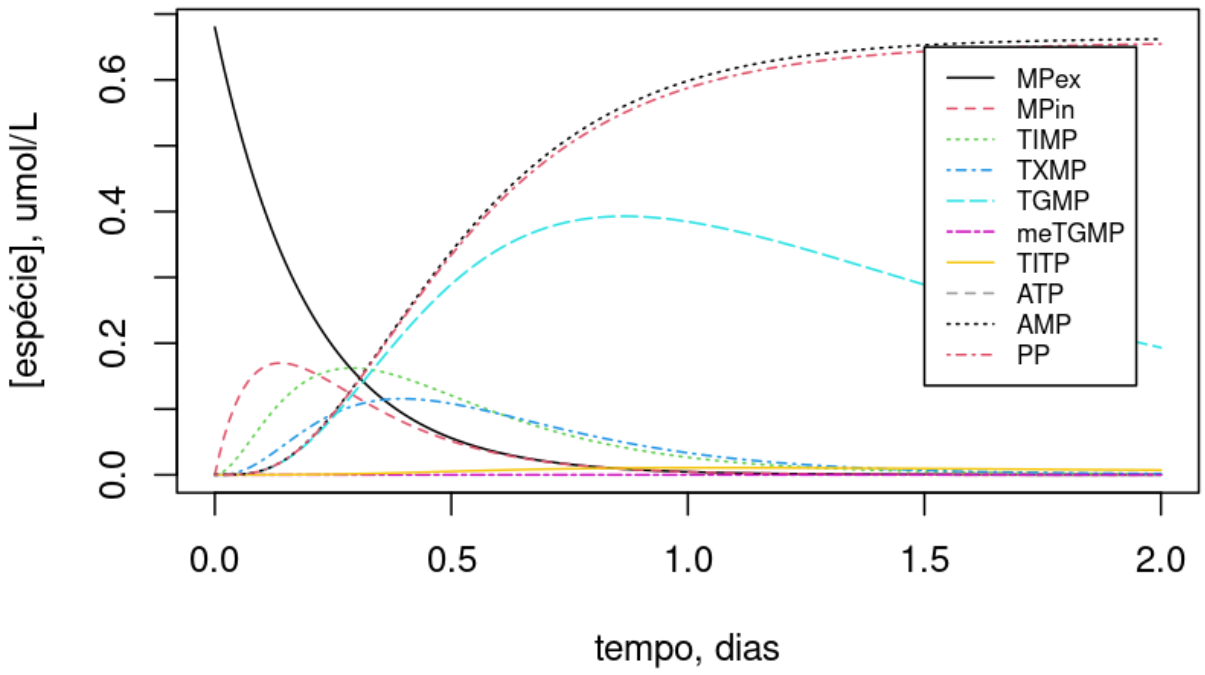


Figura 13.14: Dependência da dinâmica da rede metabólica de degradação de 6-mercaptopurina em função do teor inicial de ATP a 2 umol/L. Os valores iniciais e parâmetros são descritos no trecho de código.

Observe que, com a redução de 10x no teor de ATP, TGMP e TIMP mantiveram teores mais estáveis ao longo do período.

13

Akaike, Hirotugu. 1974. «A new look at the statistical model identification». *IEEE transactions on automatic control* 19 (6): 716–23.

Alberty, Robert A. 1991. «Chemical equations are actually matrix equations». *Journal of chemical education* 68 (12): 984.

———. 1996. «Calculation of biochemical net reactions and pathways by using matrix operations». *Biophysical journal* 71 (1): 507–15.

Bhatti, Haq Nawaz, M Hamid Rashid, Rakhshanda Nawaz, A Mukhtar Khalid, Muhammad Asgher, e A Jabbar. 2007. «Effect of aniline coupling on kinetic and thermodynamic properties of Fusarium solani glucoamylase». *Applied microbiology and biotechnology* 73 (6): 1290–98.

Bloomfield, Victor. 2009. *Computer simulation and data analysis in molecular biology and biophysics: an introduction using R*. Springer Science & Business Media.

Cooper, Alan. 2004. «Thermodynamics and interactions». Em *Biophysical Chemistry*, 99–122.

Creighton, Thomas E et al. 2010. *biophysical chemistry of nucleic acids & proteins*. Distributed by Gardners Books.

Dahlquist, FW. 1978. «[13] The meaning of scatchard and hill plots». *Methods in enzymology* 48: 270–99.

DeLean, A, PJ Munson, e Dl Rodbard. 1978. «Simultaneous analysis of families of sigmoidal curves: application to bioassay, radioligand assay, and physiological dose-response curves.» *American Journal of Physiology-Endocrinology And Metabolism* 235 (2): E97.

Duff, DG, e CH Giles. 1972. «Spectrophotometric determination of the critical micelle concentration of surfactants». *Journal of Colloid and Interface Science* 41 (3): 407–14.

Edelhoch, Harold, e James C Osborne Jr. 1976. «The thermodynamic basis of the stability of proteins, nucleic acids, and membranes». *Advances in protein chemistry* 30: 183–250.

Gandrud, Christopher. 2018. *Reproducible research with R and RStudio*. Chapman; Hall/CRC.

Johnson, Kenneth A. 1992. «1 transient-state kinetic analysis of enzyme reaction pathways». Em *The enzymes*, 20:1–61. Elsevier.

Khalil, Mutasim I. 2000. «Calculating enthalpy of reaction by a matrix method». *Journal of Chemical Education* 77 (2): 185.

Lavrova, Anastasia I, Eugene B Postnikov, Andrey Yu Zyubin, e Svetlana V Babak. 2017. «Ordinary differential equations and Boolean networks in application to modelling of 6-mercaptopurine metabolism». *Royal Society Open Science* 4 (4): 160872.

Leone, Francisco de Assis. 2021. *Fundamentos de Cinética Enzimática*. Appris Ed.

LiCata, Vince J, e Chin-Chi Liu. 2011. «Analysis of free energy versus temperature curves in protein folding and macromolecular interactions». *Methods in enzymology* 488: 219–38.

Michaelis, L., e ML Menten. 1913. «Die Kinetik der Invertinwirkung». *Biochem Z* 49 (4): 333–69.

Neto, Benicio Barros, Ieda Spacino Scarminio, e Roy Edward Bruns. 2010. *Como Fazer Experimentos-: Pesquisa e Desenvolvimento na Ciência e na Indústria*. Bookman Editora.

Otaki, Joji M, Shunsuke Ienaka, Tomonori Gotoh, e Haruhiko Yamamoto. 2005. «Availability of short amino acid sequences in proteins». *Protein science* 14 (3): 617–25.

Parsons, DL, e JJ Vallner. 1978. «Theoretical models for cooperative binding—I. one-site creator of binding sites». *Mathematical Biosciences* 41 (3-4): 189–215.

Pauling, Linus. 1935. «The oxygen equilibrium of hemoglobin and its structural interpretation». *Proceedings of the National Academy of Sciences of the United States of America* 21 (4): 186.

Po, Henry N, e NM Senozan. 2001. «The Henderson-Hasselbalch equation: its history and limitations». *Journal of Chemical Education* 78 (11): 1499.

Ross, Philip D, e S Subramanian. 1981. «Thermodynamics of protein association reactions: forces contributing to stability». *Biochemistry* 20 (11): 3096–3102.

Sarkar, Deepayan. 2008. *Lattice: multivariate data visualization with R*. Springer Science & Business Media.

Scatchard, George. 1949. «The attractions of proteins for small molecules and ions». *Annals of the New York Academy of Sciences* 51 (4): 660–72.

Spiess, Andrej-Nikolai, e Natalie Neumeyer. 2010. «An evaluation of R2 as an inadequate measure for nonlinear

models in pharmacological and biochemical research: a Monte Carlo approach». *BMC pharmacology* 10 (1): 1–11.

Tang, Lixia, Jeffrey H Lutje Spelberg, Marco W Fraaije, e Dick B Janssen. 2003. «Kinetic mechanism and enantioselectivity of halohydrin dehalogenase from Agrobacterium radiobacter». *Biochemistry* 42 (18): 5378–86.

Tong, Jianbo, Shan Lei, Shangshang Qin, e Yang Wang. 2018. «QSAR studies of TIBO derivatives as HIV-1 reverse transcriptase inhibitors using HQSAR, CoMFA and CoMSIA». *Journal of Molecular Structure* 1168: 56–64.

Traut, Thomas W. 2007. *Allosteric regulatory enzymes*. Springer Science & Business Media.

Tyuma, Itiro, Kiyohiro Imai, e Katsuhiko Shimizu. 1973. «Analysis of oxygen equilibrium of hemoglobin and control mechanism of organic phosphates». *Biochemistry* 12 (8): 1491–98.

Waelbroeck, Magali, E Van Obberghen, e P De Meyts. 1979. «Thermodynamics of the interaction of insulin with its receptor.» *Journal of Biological Chemistry* 254 (16): 7736–40.

Wickham, Hadley. 2011. «ggplot2». *Wiley interdisciplinary reviews: computational statistics* 3 (2): 180–85.

Wilkinson, GN. 1961. «Statistical estimations in enzyme kinetics». *Biochemical Journal* 80 (2): 324–32.

Yung-Chi, Cheng, e William H Prusoff. 1973. «Relationship between the inhibition constant (KI) and the concentration of inhibitor which causes 50 per cent inhibition (I50) of an enzymatic reaction». *Biochemical pharmacology* 22 (23): 3099–3108.