(Elementary) Regression Methods & Computational Statistics (405.952)

Part II: Regression (cont.)

Assoz. Prof. Dr. Wolfgang Trutschnig
Arbeitsgruppe Stochastik/Statistik
Fachbereich Mathematik
Universität Salzburg
www.trutschnig.net

Salzburg, November/December 2018

How to proceed in practice (dimensions up to $d = 3$)

- Consider the univariate case:
- Given data $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ from a model $Y = r(X) + \varepsilon$.
- r is an unknown function and ε is a random error fulfilling $\mathbb{E}(\varepsilon) = 0$.
- For sufficiently large sample size we can estimate r via kernel regression.
- If the kernel regression is of a specific form (if it looks linear, quadratic, saturation-curve-like, S-shaped, etc.) the next natural step is to fit a **parametric model**.
- We already know how to fit linear and polynomial models.
- How to proceed in the general case?
- We start with a simple example.

Example (Moisture dataset)

- The dataset moisture.txt contains moisture content of core samples from mud.
- For each sample the depth (in feet) as well as the moisture (g water per 100g dried solid).

tiefe	Wassergehalt
0.00	127.20
0.00	132.03
0.00	130.44
0.00	128.07
0.00	126.95
0.00	118.15
0.00	117.82
0.00	129.23

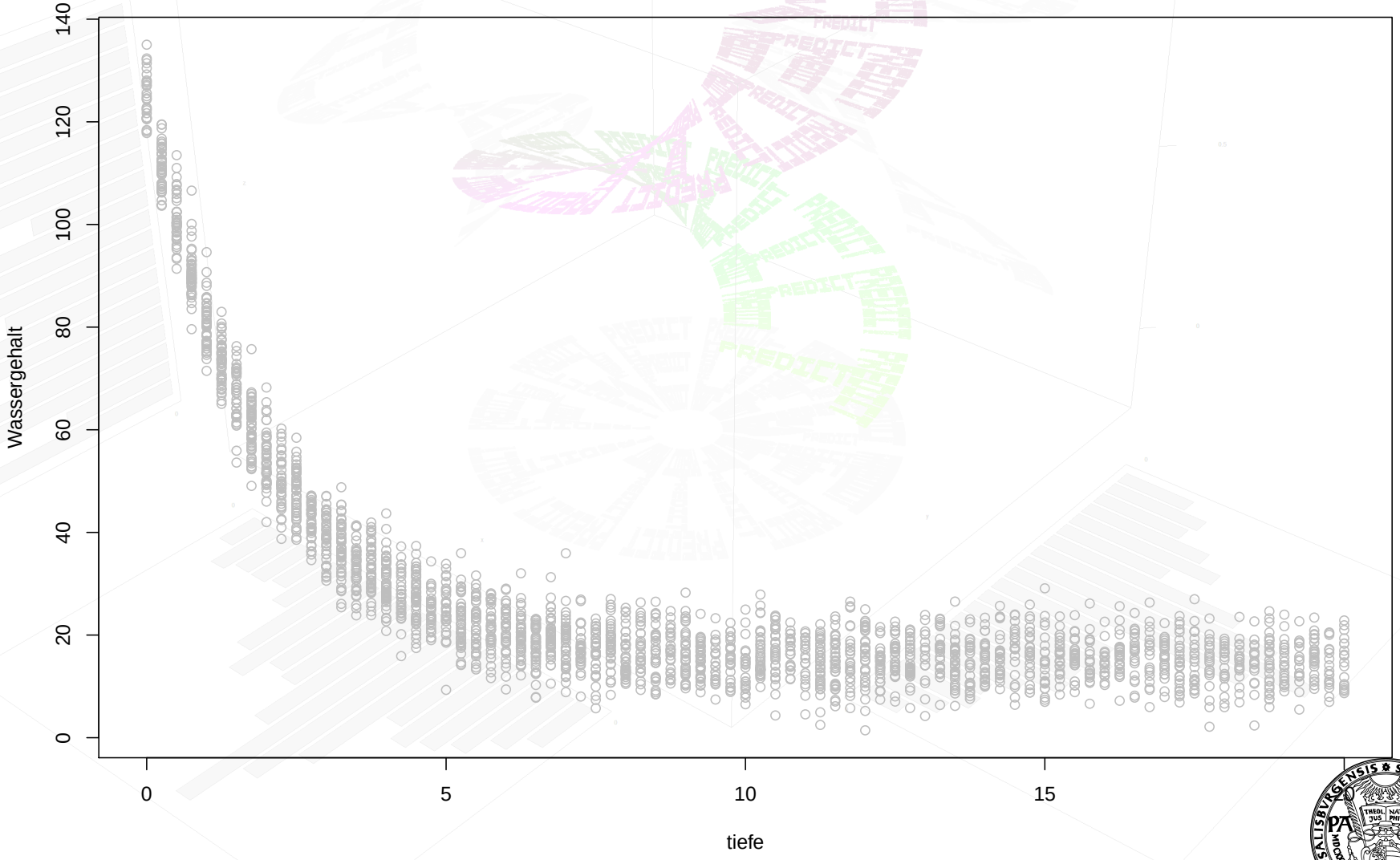
Table: First eight lines of the moisture dataset

tiefe	Wassergehalt
Min. : 0.000	Min. : 1.44
1st Qu.: 3.500	1st Qu.: 14.89
Median : 7.750	Median : 19.73
Mean : 8.522	Mean : 30.06
3rd Qu.:13.000	3rd Qu.: 33.72
Max. :20.000	Max. :135.03

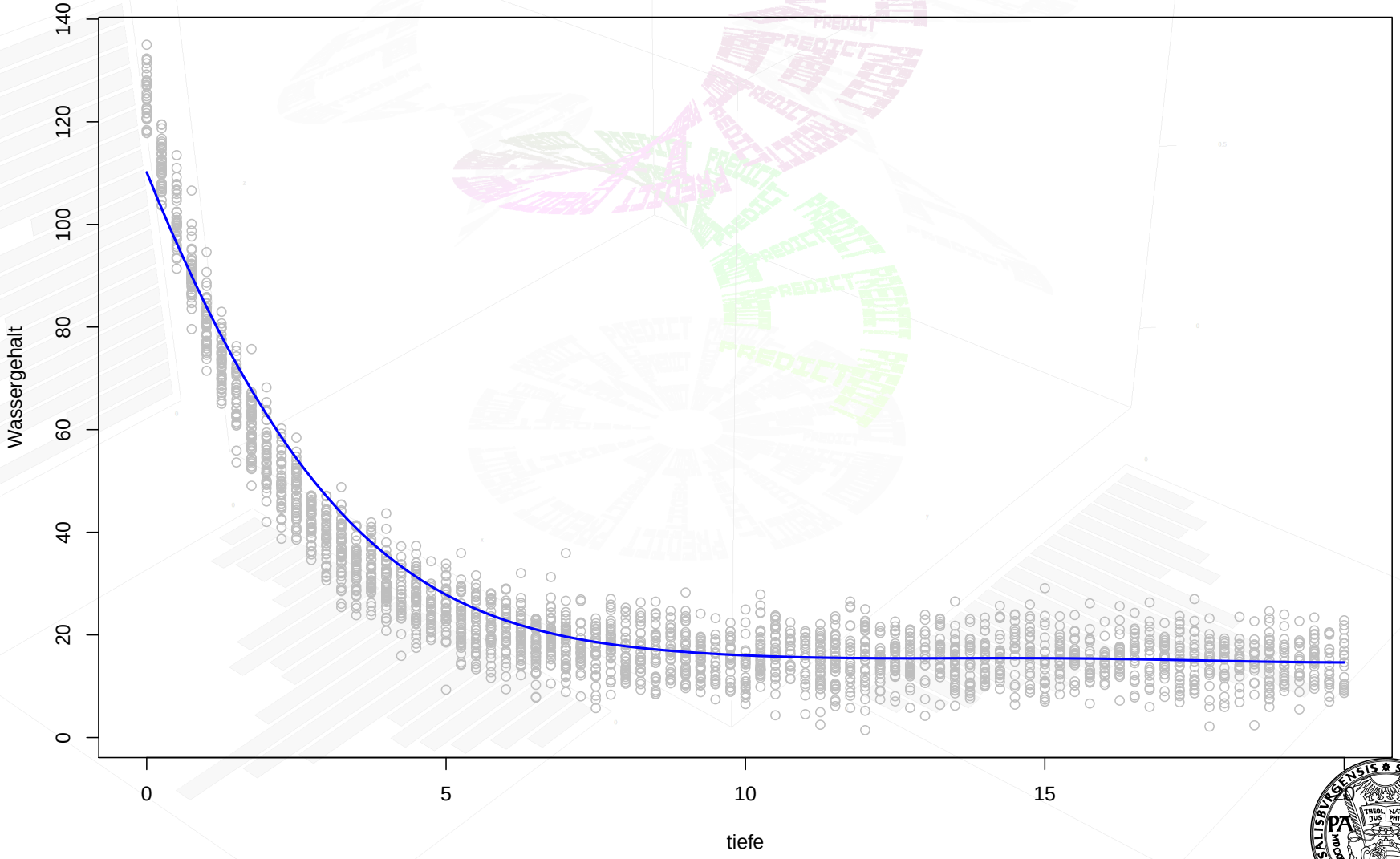
Table: Summary of the moisture dataset

- The dataset is plotted on the next slide

The function nls



The function nls



Example (Moisture dataset, continued)

- Assume that from geological models it is known that (under certain conditions) the interrelation by depth t and moisture m is given by

$$m = 125 - 110(1 - e^{-at}). \tag{1}$$

- In other words, there is a one-parametric model describing the interrelationship.
- This parameter a needs to be estimated from the observed data $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$.
- How can this be done?
- Option 1: Simply transform eq. (1) into a linear model in t.
- Option 2: Proceed as before and choose $\hat{a}$ in such a way that the sum of all squared errors

$$F(\tilde{a}) := \sum_{i=1}^{n} \left(m_i - (125 - 110(1 - e^{-\tilde{a}t_i}))\right)^2 \tag{2}$$

is minimized.

Example (Moisture dataset, continued)

- The minimization can be done using the R-function *nls*:

```
model.nls <- nls(data=A, Wassergehalt~125-110*(1-exp(-a*tiefe)),
                 start = list(a = 1))
model.nls
```

- yields

```
Nonlinear regression model
model: Wassergehalt ~ 125 - 110 * (1 - exp(-a * tiefe))
data: A
a
0.5027
residual sum-of-squares: 50676

Number of iterations to convergence: 5
Achieved convergence tolerance: 5.126e-08
```

Example (Moisture dataset, continued)

- Moreover

```
summary(model.nls)
```

- yields

```
Formula: Wassergehalt ~ 125 - 110 * (1 - exp(-a * tiefe))

Parameters:
   Estimate   Std. Error t value Pr(>|t|)
a 0.502668   0.002241   224.3   <2e-16 ***
  ---
Signif. codes:  0   ***   0.001   **   0.01   *   0.05
    .   0.1        1

Residual standard error: 4.493 on 2510 degrees of freedom

Number of iterations to convergence: 5
Achieved convergence tolerance: 5.126e-08
```

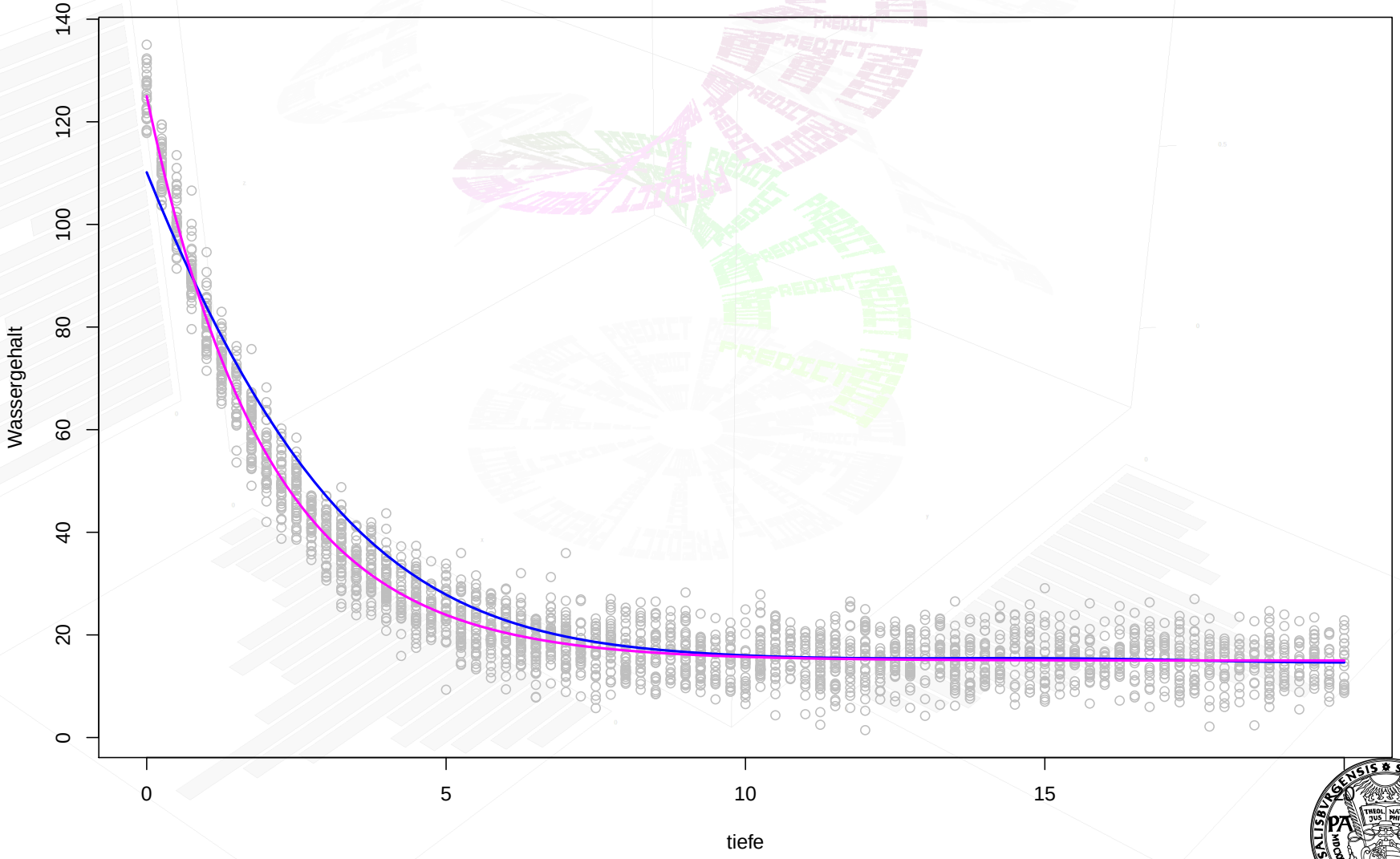
Example (Moisture dataset, continued)

- We add the function $m(t) = 125 - 110(1 - e^{-at})$ for the estimated parameter $a = 0.502668$ to the plot.

```
new <- data.frame(tiefe=seq(0,20,length=1001))
pred <- predict(model.nls,newdata = new)
new$prediction <- pred
lines(new$tiefe,new$prediction,col="red",type="l")
```

- Notice that the syntax is exactly the same as for *lm*, only the command itself is replaced by *nls* and a reasonable starting value (initial guess) for *a* has to be provided.
- The following graphic shows the sample as well as the kernel regression (blue) and the parametric regression calculated via nls (magenta).

The function nls



Example (SBP versus age and BMI)

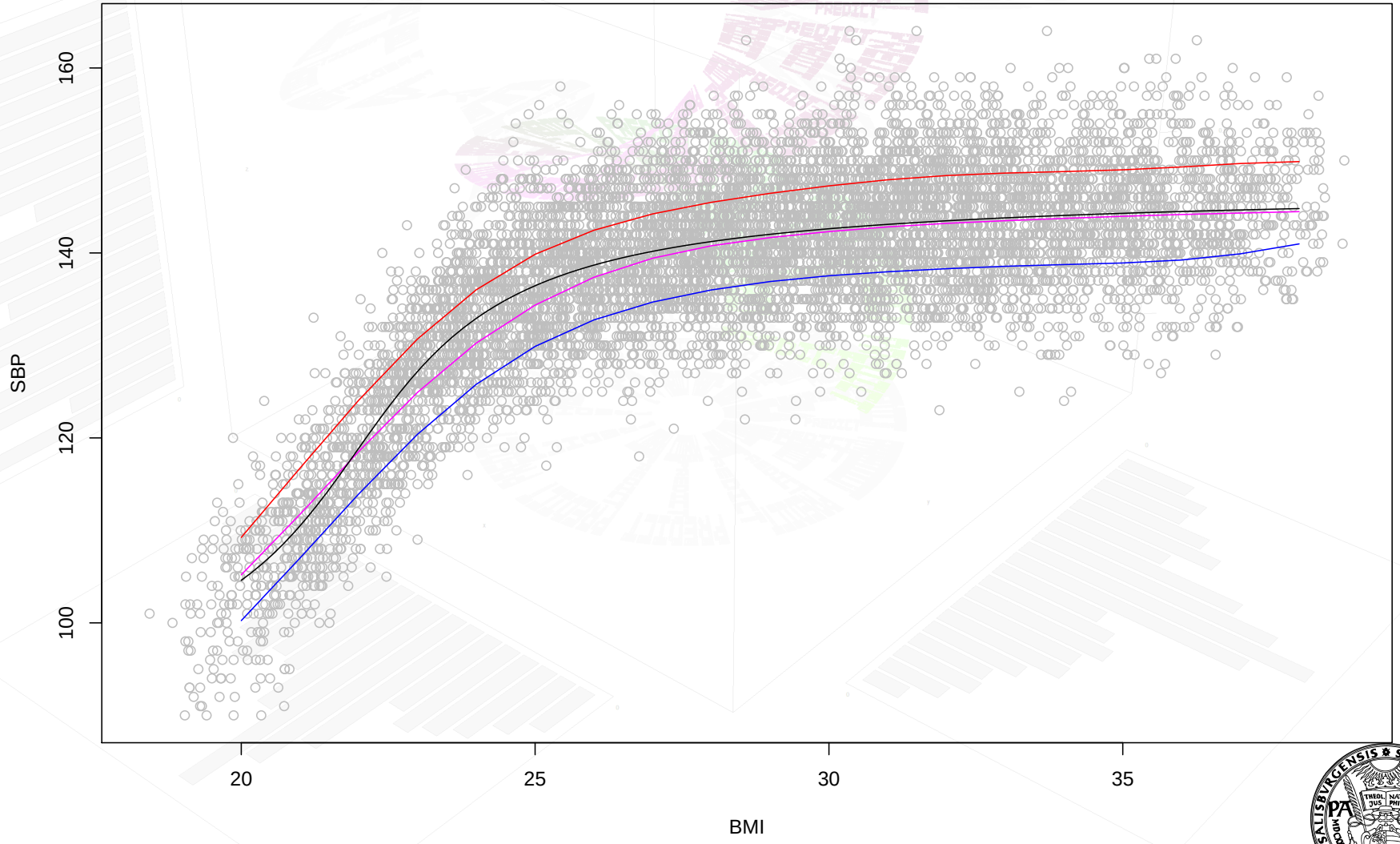
- The data set SBP.RData (see Exercise 13) contains the following data for 8.000 patients: age, BMI (body mass index), SBP (systolic blood pressure).

age	BMI	SBP
61.00	26.05	131.00
37.00	26.47	137.00
77.00	36.35	151.00
75.00	25.75	147.00
39.00	28.20	141.00
34.00	25.32	129.00

Table: First six lines of the SBP dataset

- In Exercise 13 we considered three age groups: $[30, 40]$, $[50, 60]$ and $[70, 80]$.
- For each age group we calculated the kernel regression estimate for the regression function r with $SBP = r(BMI)$ and got the following result:

The function nls, cont.



- The function *nls* can be used in arbitrary dimensions (the reason for the restriction on dimension 1-3 before was due to the kernel regression)

Example (SBP versus age and BMI, continued)

- We fit a regression model of the form

$$SBP = r(BMI, age) = 105 + a \cdot age + b \cdot atan(c \cdot BMI + d)$$

- The model contains four parameters: a, b, c, d

```
model.nls <- nls(data=A, SBP ~ 105+a*age + b*atan(c*BMI + d),
                 start = list(a=1, b=10, c=0.5, d=-5))
model.nls
summary(model.nls)
```

yields

Example (SBP versus age and BMI, continued)

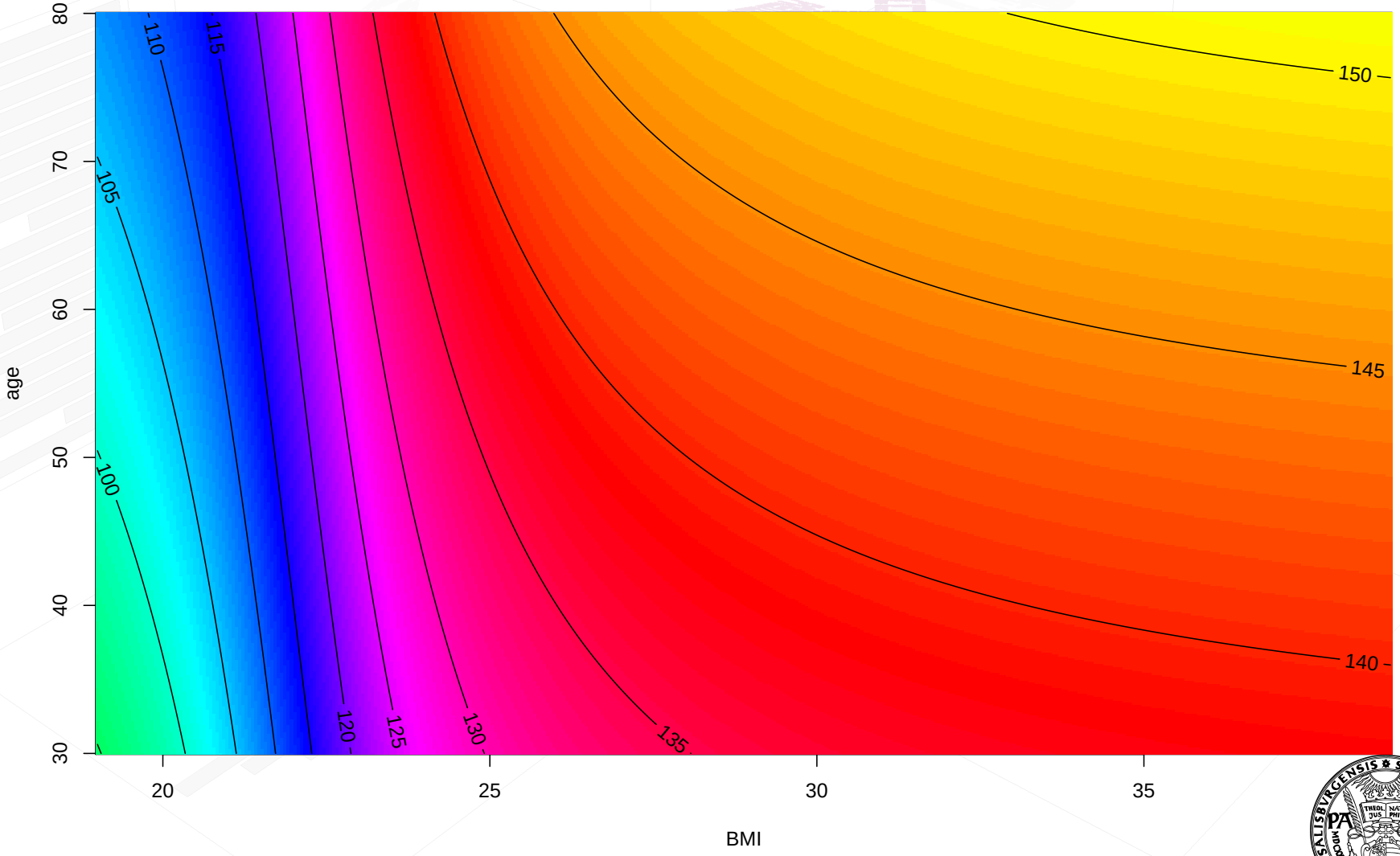
- ```
  Parameters:
        Estimate   Std. Error t value Pr(>|t|)
  a     0.252004   0.003506   71.87   <2e-16 ***
  b    17.830446   0.178180  100.07   <2e-16 ***
  c     0.510601   0.010566   48.33   <2e-16 ***
  d   -11.236677   0.240598  -46.70   <2e-16 ***
  ---
  Signif. codes:  0  ***  0.001  **  0.01  *  0.05
  .  0.1  1
  ```

- as well as

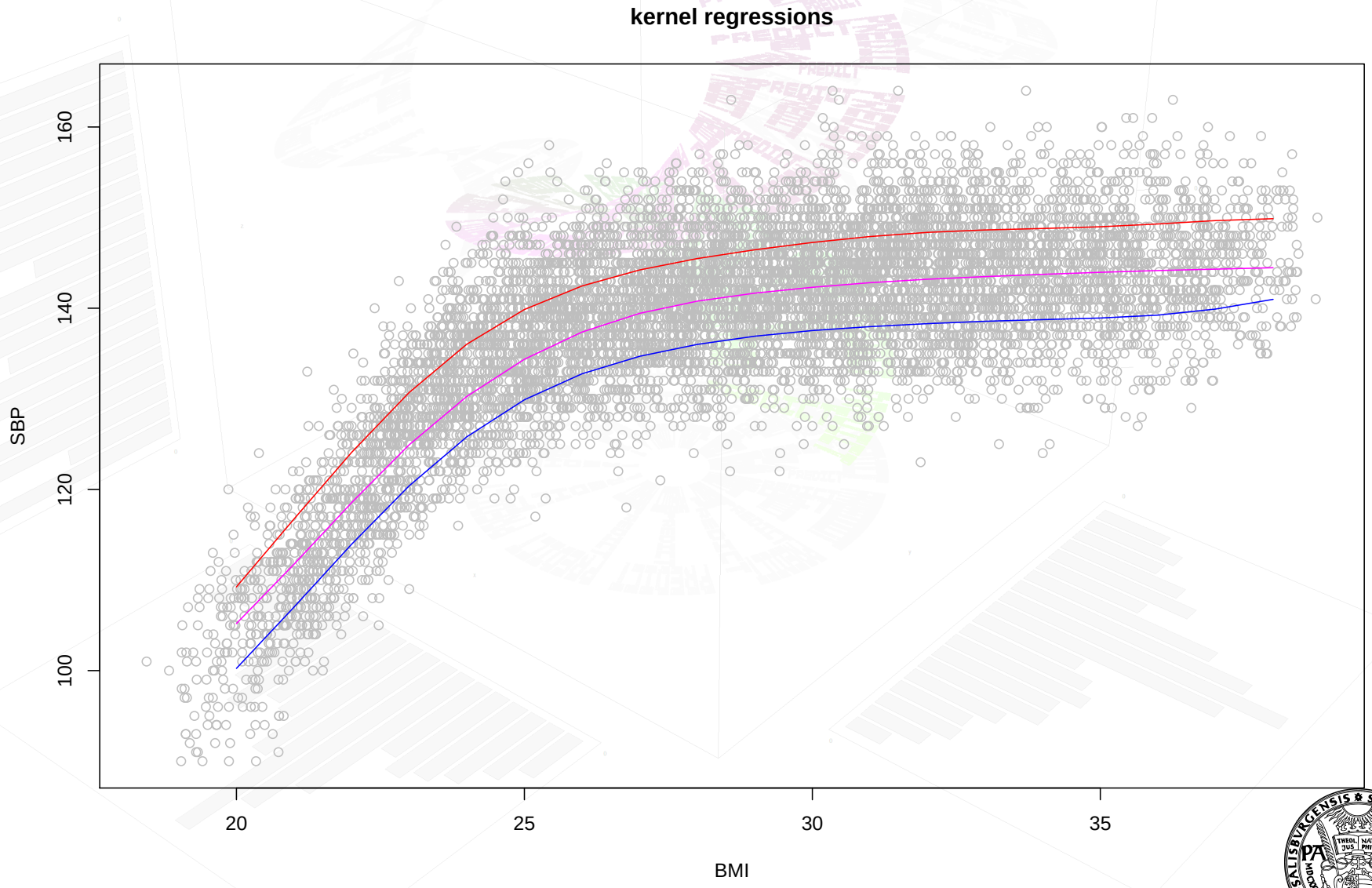
```
Number of iterations to convergence: 6
Achieved convergence tolerance: 5.443e-07


#image plot of the regression function
pars <- as.numeric(coef(model.nls))
reg <- function(BMI,age){
  r <- 105 + pars[1]*age + pars[2]*atan(pars[3]*BMI + pars[4])
  return(r)
```
```

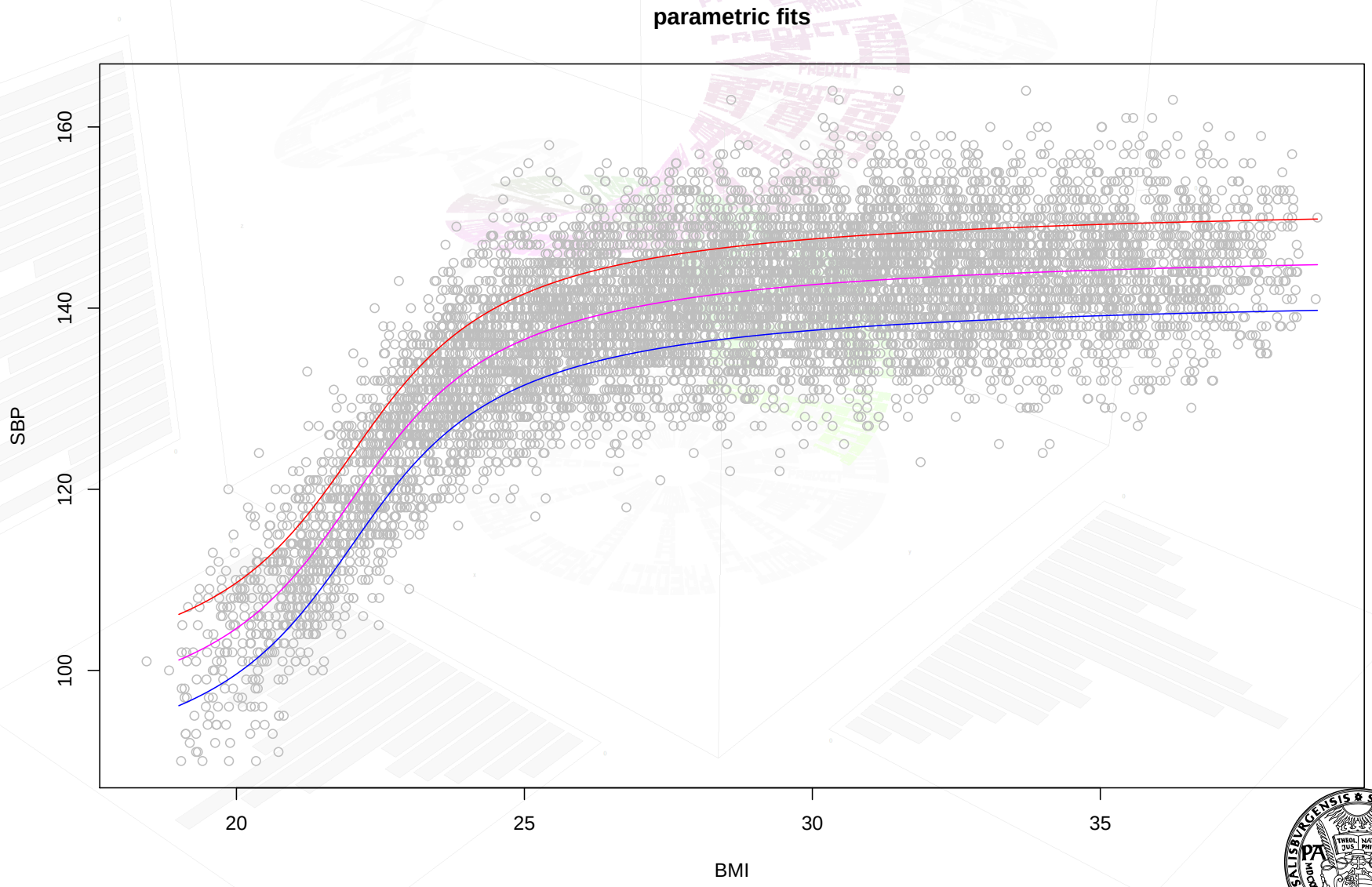
The function nls, cont.



The function nls, cont.



The function nls, cont.



A word of caution

- *nls* tries to find the absolute minimum of a function based on numerical methods.
- Hence good starting values (initial guesses) for the parameters are important - otherwise the algorithm might run into local minima or even produce no results (error message "Singular gradient matrix at initial parameter estimates").
- It often helps to first try out some parameter values and calculate the sum of the squared residuals F and then run *nls* with those initial parameters which resulted in the smallest value of F.

Outlook:

- Sometimes the parameters also have to fulfill certain boundary conditions.
- In this case the *optim* function can be used .

Exercise 14:

- The data set *beer.txt* contains foam height at various time points for three brands of beer.
- Use R-Codes_Regression06.R to load the data.
- Fit an individual regression model of the form

$$H = H_0 \, e^{-at}$$

  to the data of each brand. Thereby H denotes the foam height, t the time (in seconds), and H_0 and a are parameters.
- According to the models, which brand is expected to have the highest foam at time point $t = 480$?
- Include graphics, the resulting regression models as well as the answer to the afore-mentioned question in a short knitR report.

Exercise 15:

- Write a knitR report analyzing the *Prestige* dataset using the tools we have learned so far.
- Use the last lines in R-Codes_Regression06.R to load the data and understand what the dataset contains.
- Come up with some conjectures (which variables might have an influence on prestige scores, etc.?) and answer them using regression and descriptive statistics.

Example (Passing an exam versus hours of study; taken from wikipedia)

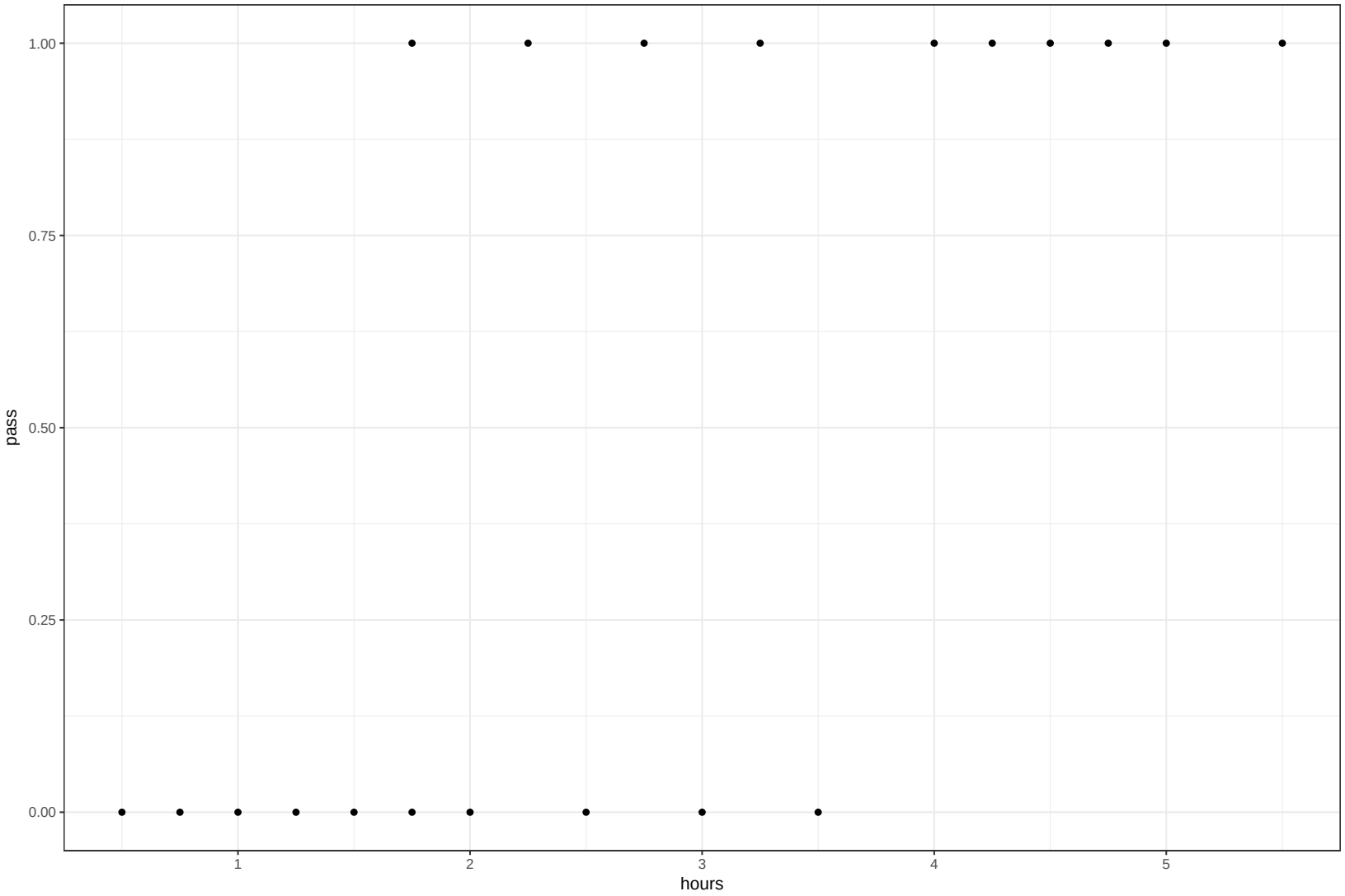
- A group of 20 students spend between 0 and 6 hours studying for an exam.
- How does the number of hours spent studying affect the probability that the student will pass the exam?

	1	2	3	4	5	6	7	8	9	10
hours	0.50	0.75	1.00	1.25	1.50	1.75	1.75	2.00	2.25	2.50
pass	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	1.00	0.00

	11	12	13	14	15	16	17	18	19	20
hours	2.75	3.00	3.25	3.50	4.00	4.25	4.50	4.75	5.00	5.50
pass	1.00	0.00	1.00	0.00	1.00	1.00	1.00	1.00	1.00	1.00

- What is the difference to the models of the form $y = r(x) + \varepsilon$ considered so far?

Logistic regression - introductory example



Example (Passing an exam versus hours of study cont.)

- The dependent variable 'pass' is binary - it only attains the values 0 (fail) and 1 (pass).
- Call Y (=pass) the dependent and X (=hours) the explanatory variable.
- We could formalize the model as follows

$$\mathbb{P}(Y = 1|X = x) = r(x),$$

  i.e. the probability of passing the exam after having studied x hours is given by $r(x) \in [0, 1]$.
- In other words: Y given $X = x$ has a Bernoulli (Binomial) distribution with parameter $p = r(x)$.
- How to estimate r?

Example (Passing an exam versus hours of study cont.)

- One standard choice is to work with a parametrization of the so-called logistic functions: Set

$$r(x) = \frac{1}{1 + e^{-ax-b}}$$

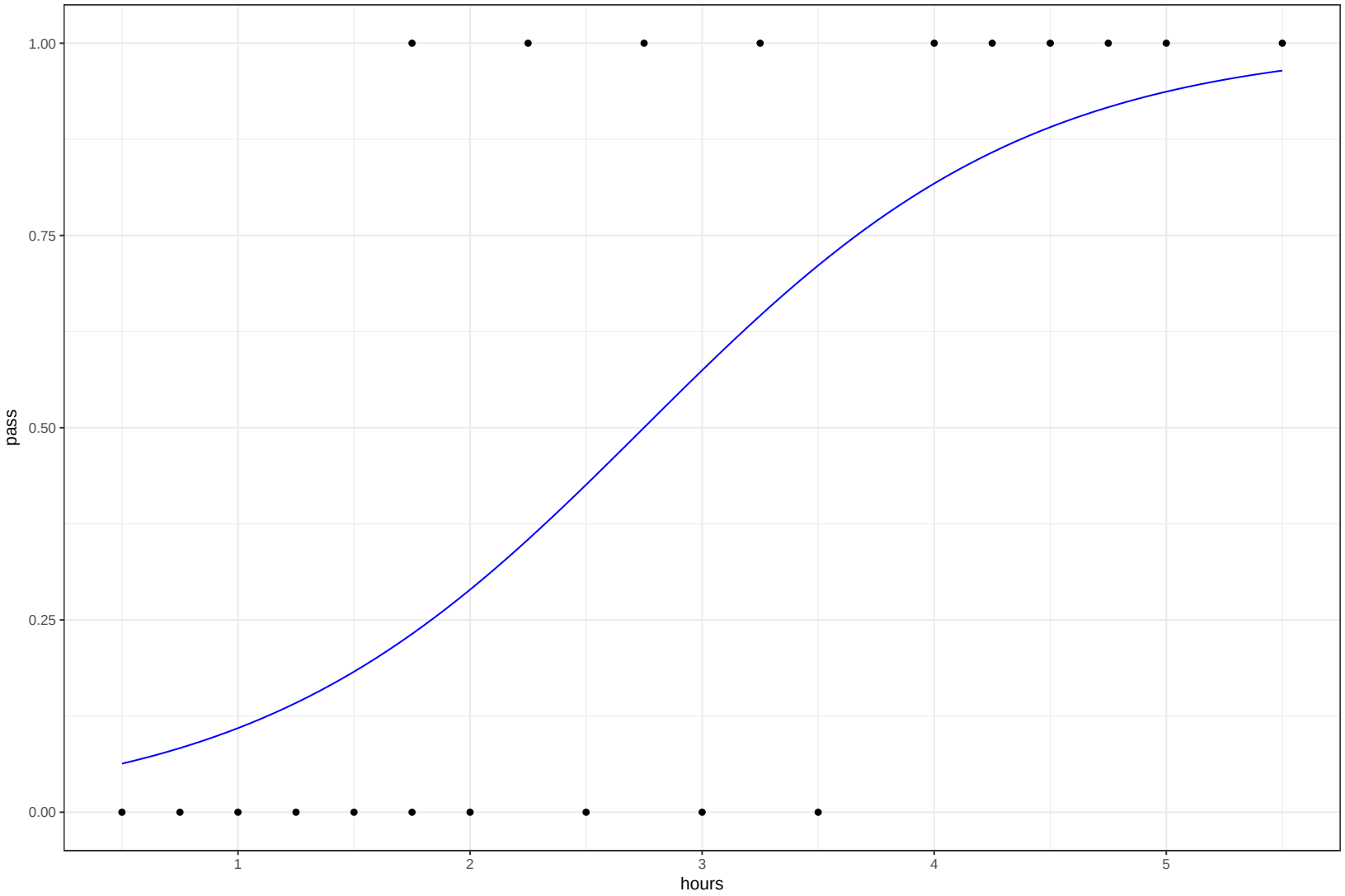
and then try to find the best choice (in the least squares sense) for the parameters $a, b \in \mathbb{R}$.

- Doing so we could use nls as in the previous section:

```
model.nls <- nls(data=A, pass ~ 1/(1+exp(-a*hours-b)), start =
    list(a=0,b=1))
model.nls
```

```
Nonlinear regression model
model: pass ~ 1/(1 + exp(-a * hours - b))
data: A
a      b
1.199  -3.296
residual sum-of-squares: 2.69
```

Logistic regression - introductory example



Logistic regression - introductory example

- The standard procedure for estimating a and b, however, is different.
- The standard is to optimize the likelihood, i.e. (in this setting) the probability to observe what he observed.
- Suppose that the sample $(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$ has been observed.
- We proceed in several simple steps:
- If $y_1 = 1$ then $\mathbb{P}(Y_1 = 1 | X = x_1) = r(x_1)$.
- If $y_1 = 0$ then $\mathbb{P}(Y_1 = 0 | X = x_1) = 1 - r(x_1)$.
- Altogether we get: $\mathbb{P}(Y_1 = y_1 | X = x_1) = r(x_1)^{y_1}(1 - r(x_1))^{1-y_1}$.
- Proceeding in the same manner (and assuming independence) we get the following Likelihood function ℓ:

$$\ell\big((x_1, y_1), \ldots (x_n, y_n); a, b\big) = \prod_{i=1}^{n} r(x_i)^{y_i}(1 - r(x_i))^{1-y_i}$$

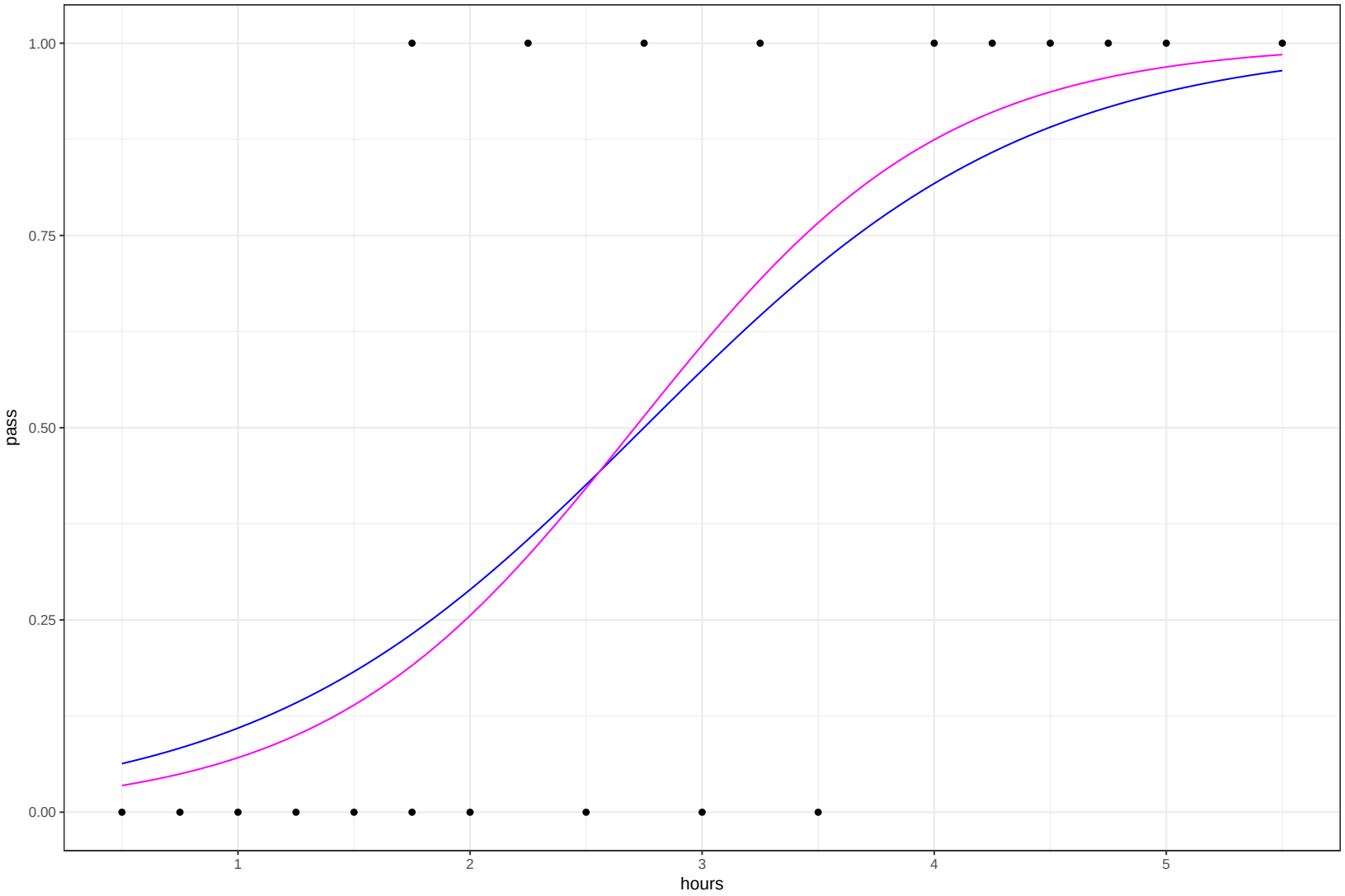
Logistic regression - introductory example

- The Likelihood function ℓ is then maximized with respect to the parameters a, b.
- No need to calculate ℓ and use nls - directly use glm in R:
- ```
  model.glm <- glm(data=A, pass~hours, family = "binomial")
  model.glm
  ```
- yields
- ```
  Call:  glm(formula = pass~hours, family = "binomial", data = A)

  Coefficients:
    (Intercept)        hours
  -4.078        1.505

  Degrees of Freedom: 19 Total (i.e. Null);  18 Residual
  Null Deviance:      27.73
  Residual Deviance: 16.06   AIC: 20.06
  ```

Logistic regression - introductory example



Logistic regression - introductory example

- Why didn't we also maximize the likelihood in linear regression?

- Which of the two estimates works better?
- Let's run a simulation study and check the performance of both estimators.
-

```
p03<-p


n <- 100
a <- 1.5; b <- (-4)
x <- runif(n,0,5)
y <- rep(0,n)
for(i in 1:n){
  p <- 1/(1+exp(-a*x[i]-b))
  y[i] <- sample(c(0,1),prob=c(1-p,p),size=1)
}
A <- data.frame(x=x,y=y)
model.nls <- nls(data=A, y ~ 1/(1+exp(-a*x-b)), start = list(a
    =0,b=1))
```

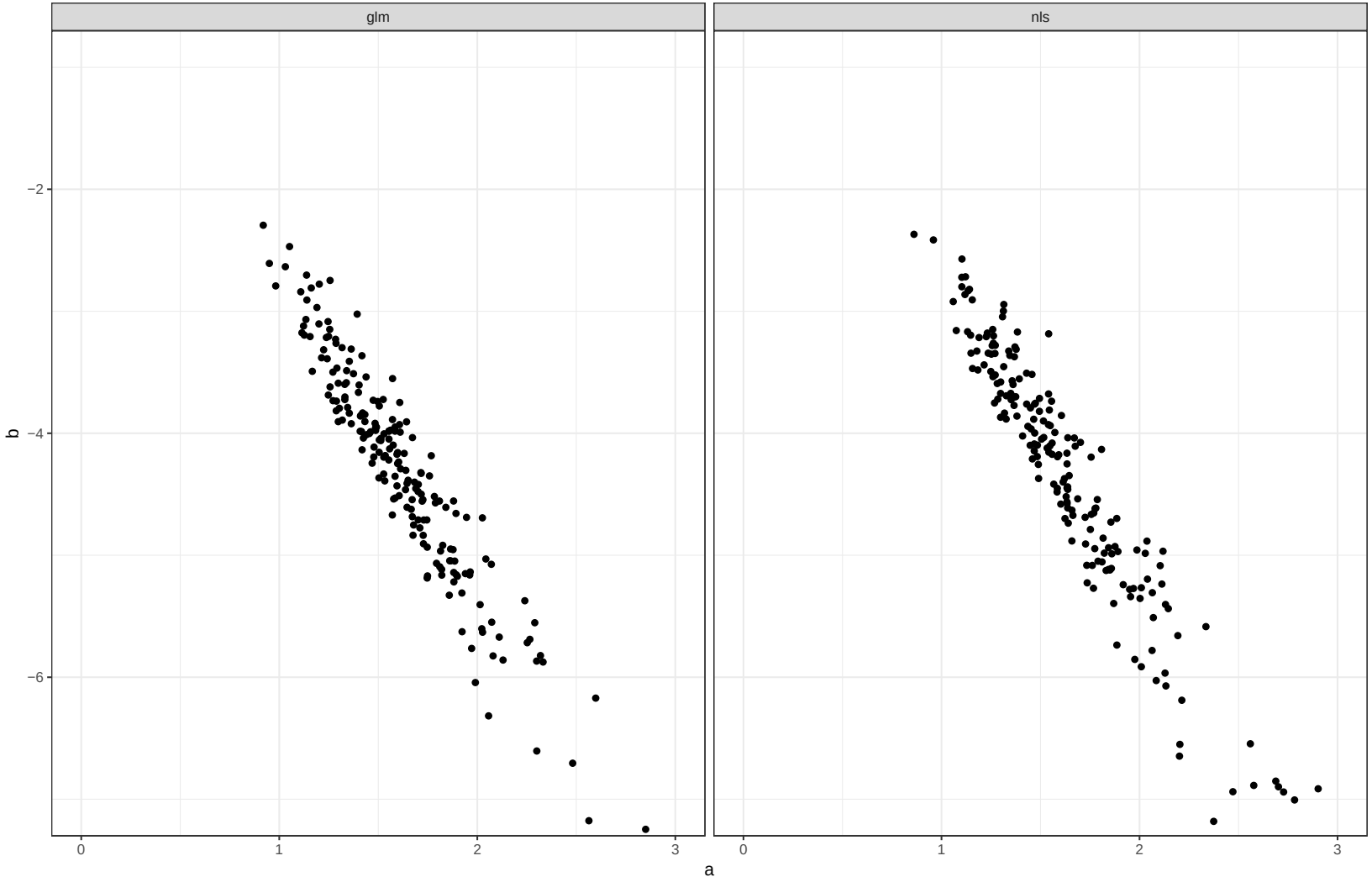
```r
model.glm

#systematic
R <- 200
E.nls <- data.frame(a=rep(0,R),b=rep(0,R))
E.glm <- data.frame(a=rep(0,R),b=rep(0,R))

for(j in 1:R){
  n <- 100
  a <- 1.5; b <- (-4)
  x <- runif(n,0,5)
  y <- rep(0,n)
  for(i in 1:n){
    p <- 1/(1+exp(-a*x[i]-b))
    y[i] <- sample(c(0,1),prob=c(1-p,p),size=1)
  }
  A <- data.frame(x=x,y=y)
  model.nls <- nls(data=A, y ~ 1/(1+exp(-a*x-b)), start = list(a
      =0.5,b=-3), control=list(maxiter = 500))
  E.nls[j,1:2] <- as.numeric(coefficients(model.nls))
  model.glm <- glm(data=A, y ~ x, family = "binomial")
  E.glm[j,1:2] <- as.numeric(coefficients(model.glm)[2:1])
```

Performance

Estimates within the chosen window [0,3]x[−7,−1]



Exercise 16:

- Download the full titanic dataset as xls from here.
- Import the data in R.
- Find out if the variable 'age' had an influence on the survival probability (via fitting a logistic regression).
- What about the variable 'fare'?

Exercise 17:

- Figure out how multivariate logistic regression can be done in R.
- Find a suitable dataset (there are many available online) or provide an own dataset where logistic regression seems suitable.
- Fit a multivariate logistic regression to the data.

Done so far:

- Given samples $(x_1, y_1), \ldots, (x_n, y_n)$ from a model $y = r(x) + \varepsilon$ we tried to estimate the regression function r.
- With the estimate $\hat{r}$ the prediction at the point x is then given by $\hat{r}(x)$.
- For the linear case $r(x) = ax + b$ (and some others) we have verified via simulations that for large sample size n the estimates $\hat{a}_n, \hat{b}_n$ were very close to the true values a, b (increasing the sample size had a 'zooming in' effect).
- If the estimates of the parameters are good, then predictions will be good.
- Will the (expected) prediction error be the same for all values of x?
- So far we have only considered point estimates but no confidence intervals, neither for the parameters nor for the predictions.
- $\rightarrow$ plan for next week.
- Additionally: Can we expect the predictions calculated for new data to be as good as for the data at hand (i.e. the data to which we fitted the linear model)?
- $\rightarrow$ we first have a look to cross-validation (= one of the options to handle this problem) and start with simulations.

Basic idea (in the context of linear regression)

- In general it seems natural that the predictions are better for the original sample than for the new data (why?).
- A natural way to quantify the expected prediction quality of a model is the so-called *mean-squared-error* (MSE, for short).
- We have already seen the MSE implicitly: 'sum of squared residuals'.
- Expressed mathematically: $(x_1, y_1), \ldots, (x_n, y_n)$ sample from a model $y = r(x) + \varepsilon$, $\hat{a}_n$ and $\hat{b}_n$ estimates of the parameters a and b.
- Suppose we are given new data $(x_1', y_1'), \ldots, (x_n', y_n')$ from the same model (and with x-values from the same population as $x_1, \ldots, x_n$).
- Then, using the estimates $\hat{a}_n$ and $\hat{b}_n$ (calculated from the original data), the MSE is given by

$$\frac{1}{n} \sum_{i=1}^{n} (y_i' - \hat{a}_n x_i' - \hat{b}_n)^2$$

- For the original sample the MSE is given by

$$\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{a}_n x_i - \hat{b}_n)^2$$

- R-Codes_Regression07 contains the following example illustrating this *overfitting* effect:

```
n <- 20
x <- runif(n,0,1)
eps <- rnorm(n,0,1)
y <- 2*x + eps
A <- data.frame(x=x,y=y)
model <- lm(data=A,y~x)

A$prediction <- predict(model,newdata = A)
mse <- mean((A$prediction-A$y)^2)

x.new <- runif(n,0,1)
eps <- rnorm(n,0,1)
y.new <- 2*x.new + eps
A.new <- data.frame(x=x.new,y=y.new)
A.new$prediction <- predict(model,newdata = A.new)
mse.new <- mean(abs(A.new$prediction-A.new$y)^2)
mse;mse.new
[1] 0.7111612
[1] 0.929324
```

- Notice that in the R-Code on the last slide the x-values from the original and the new sample were from the uniform distribution on $[0, 1]$ - why?
- Only if the x-values come from the same distribution the MSE-values are comparable (remember: the prediction quality is not the same for all x-values).
- Check if we only had bad luck or if the in.sample MSE really tends to be smaller than the out.sample MSE.

Basic idea (in the context of linear regression)

```r
#repeat several times and save the two mses
R <- 1000
MSE <- data.frame(mse.in=rep(0,R),mse.out=rep(0,R))
for(i in 1:R){
  n <- 20
  x <- runif(n,0,1)
  eps <- rnorm(n,0,1)
  y <- 2*x + eps
  A <- data.frame(x=x,y=y)
  model <- lm(data=A,y~x)

  A$prediction <- predict(model,newdata = A)
  MSE$mse.in[i] <- mean((A$prediction-A$y)^2)

  x.new <- runif(n,0,1); eps <- rnorm(n,0,1)
  y.new <- 2*x.new + eps
  A.new <- data.frame(x=x.new,y=y.new)
  A.new$prediction <- predict(model,newdata = A.new)
  MSE$mse.out[i] <- mean((A.new$prediction-A.new$y)^2)
}

plot(MSE)
abline(a=0,b=1)
MSE$worse <- ifelse(MSE$mse.out>=MSE$mse.in,1,0)
summary(MSE)
```

- Basic idea: Get rid of this in.sample and out.sample effect by splitting the data at hand into two sets: The *training sample* and the *validation sample*.
- Fit the model to the training sample, then predict for the validation sample.
- Should yield a realistic estimate for the prediction quality of the model.

Standard approach:

- **10-fold cross validation** (in the context of a linear model):
  - Randomly partition the data into 10 groups of equal size.
  - Ignore the data in group 1, fit a linear model to the data in the remaining 9 groups. Use this model to predict the values for the data in group 1, calculate the MSE of these predictions and save the value.
  - Proceed in the same way with the data in group 2, save the MSE.
  - Proceed in the same way with the data in group 3, save the MSE.
  - ...
  - Proceed in the same way with the data in group 10, save the MSE.
  - Average the resulting 10 values and use them as an estimate for the MSE of the model.

General k-fold cross-validation

- Randomly partition the data into k groups of equal size.
- Ignore the data in group 1, fit a linear model to the data in the remaining $k-1$ groups. Use this model to predict the values for the data in group 1, calculate the MSE of these predictions and save the value.
- Proceed in the same way with the data in group 2, save the MSE.
- ...
- Proceed in the same way with the data in group k, save the MSE.
- Average the resulting k values and use them as an estimate for the MSE of the model.

Exercise 18:

- Implement the 10-fold-cross validation described before.
- Start with a sample of size $n = 100$ from the model $y = 2x + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, 1)$ and x-values randomly chosen from a uniform distribution on $[0, 10]$.
- Save all 10 values of the MSE in a vector and compare them with the expected mean squared error obtained by changing the sample size in R-Codes on the previous slide to $n = 100$.

Exercise 19:

- Do a simulation study to answer the following questions regarding linear regression:
- Can we get better estimates than the standard ones (i.e. the estimates calculated for the full sample) if we work with 10-fold cross-validation and choose the parameter estimates (out of the 10 we get) which yield the smallest MSE?
- Work with a sample of size $n = 100$ from the model $y = 2x + 1 + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, 1)$ and x-values randomly chosen from a uniform distribution on $[0, 10]$.
- Embed your simulation study in a knitR-report summarizing the main observations.

Exercise 20:

- Cross validation may also be helpful in the context of logistic regression.
- In this case the MSE is replaced by the misclassification rate (i.e. the percentage of wrong predictions)[1]
- Use R-Codes_Regression06.R and R-Codes_Regression07.R to implement a 10-fold cross-validation .
- Save all 10 values of the misclassification rate in a vector and compare them with the expected misclassification rate (you will have to think about how the latter can be calculated).

[1] In logistic regression we predict either 1 (if $r(x) >= 0.5$) or 0 (otherwise).