Vignette zum Paket OlfaBA

ULR

2020-02-18

Das Paket BTBA1

Diese Paket soll als Langzeitalternative die Studenten im Studiengang Biotechnologie vor der Verwendung von Office-Programmen lösen. Laden könnt ihr das Paket über:

```
load_all()
#> Loading OlfaBA
```

Grundlegendes

Merkmale

Bei Studenten wird zur Qualitätsbewertung von Dozenten eigentlich immer nur zwischen "gut" oder "schlecht" unterscieden, so wie wir Farben als "grün" oder "rot" usw. bezeichnen, es gibt keine zwischengelagerten Zustände. Sollte eine solche Einteilung vorgenommen werden, so liegt eine Nominalskala vor. Das Gegenteil ist die Ordinalskala, sie kennt beliebig viele diskrete Zustände wie wir sie zum Beispiel am pH-Meter während einer Titration ablesen können. Alternativ kann man natürlich auch beim Dozente-Ranking Punkte und sogar Punkte mit Nachkommastellen vergeben. Abängig von diesen Merkmalen stehen uns verschiedene Möglichkeiten zur Verfügung, was im Anschluss mit unseren Messdaten geschieht und wie sie zu behandeln sind. Die Intervallskala hingegen erlaubt es die Differenzen auf der Ordinalskala (Intervalle) miteinzubeziehen, also eine Merkmalsausprägung. Zuletzt bleibt die Verhälnisskala, welche wie ihr Name verrät, Verhälnisse miteinbezieht und somit einen Vergleich unterschiedlicher Messungen erlaubt (*Covarianz* als Beispiel).

Klassen und die Klassenbreite

In der Biotechnologie ist meist bekannt welcher Klasse eine Stichprobe angehört, die Klassen sind dabei die übergeordneten Gruppen welche wiederum Merkmalswerte enthalten welche in den einzelnen Stichproben wiederzufinden sind. So als Beispiel eine Mehrfachbestimmung aus welcher dann eine Vielzahl technischer Replikate entsteht. Bei einer Absorptionsspektroskopischen Untersuchung zum Beispiel mehrfach Messungen mit gleichen Konzentrationen durchgeführt wobei immer Absorptionen gemessen werden, welche sich im Idealfall, kaum unterscheiden. Die Klasse ist somit schon im Voraus klar. Eine populäre Methode zur Berechnung der Klassenbreite b ist Sturges-Regel (Sturges 1926) unter Berücksichtigung des Stichproben-Umfangs n und der Spannweite v:

$$b = \frac{v}{1 + 3.32 \cdot \log n} \approx \frac{v}{5 \log n}$$

oder nach Scott (Scott 1979) unter Berücksichtigung der Standardabweichung σ :

$$b = \frac{3{,}49 \cdot \sigma}{\sqrt[3]{n}}$$

Mittelwert

Innerhalb einer solchen Klasse wird dann im eben beschriebenen Beispiel der Mittelwert errechnet. In R geschieht dies über die Funktion *mean()*. Eine Funktion kann aufgefasst werden als "Miniprogramm". Damit dieses Miniprogramm läuft braucht es Informationen, diese bekommt es vom Anwender. Diese Information ist, bezogen auf die Berechnung des arithmetischen Mittels, eine gewisse Anzahl von Werten, wobei für die Funktion *mean()* dabei schon alle wichtigen Information vorhanden sind, nämlich den Stichprobenumfang n sowie die gemessenen Werte x_i:

$$\overline{n} = \frac{1}{n} \sum x_i$$

Hierzu verwenden wir die Funktion *c()*, sie fügt die einzelnen Werte zu einem Vektor zusammen.

```
mean(c(1, 2, 3, 4))
#> [1] 2.5
# oder auch:
a <- c(1, 2.44, .56, 0.33, 1234)
mean(a)
#> [1] 247.666
```

Wichtig ist zu beachten dass die Zahlen durch ein Komma getrennt werden und Nachkommastellen durch eine Punkt! Der Pfeil ($<-$) dient als Zeichen zur Zuordnung eines Objektes, wie unserem Vektor zu einer Variablen, unser a. Eine Zuweisung in eine Andere Richtung ist unzulässig ($->$) So können wir auch gleich eine Zuweisung des Mittelwert zu vornehmen:

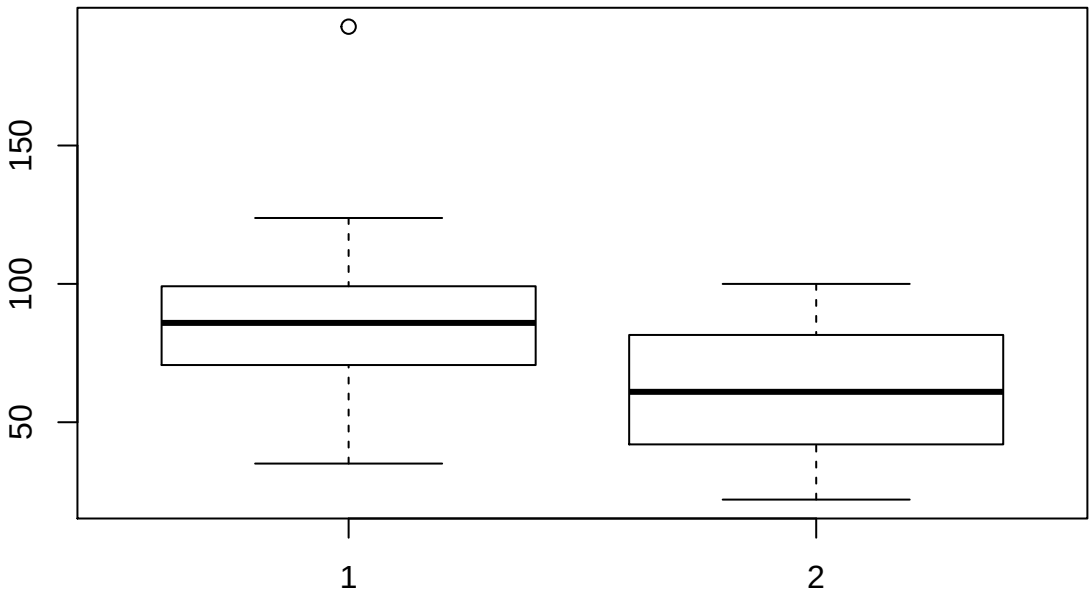
```
Mittelwert <- mean(a)
print(Mittelwert)
#> [1] 247.666
(Mittelwert)
#> [1] 247.666
```

Hier verwenden wir die Funktion *print()* um gleich den Mittelwert auf die Konsole gepromtet zu bekommen, das geschieht auch wenn wir die Variable *Mittelwert* umklammern. So auch wenn wir eine ganze Rechenoperation umklammern:

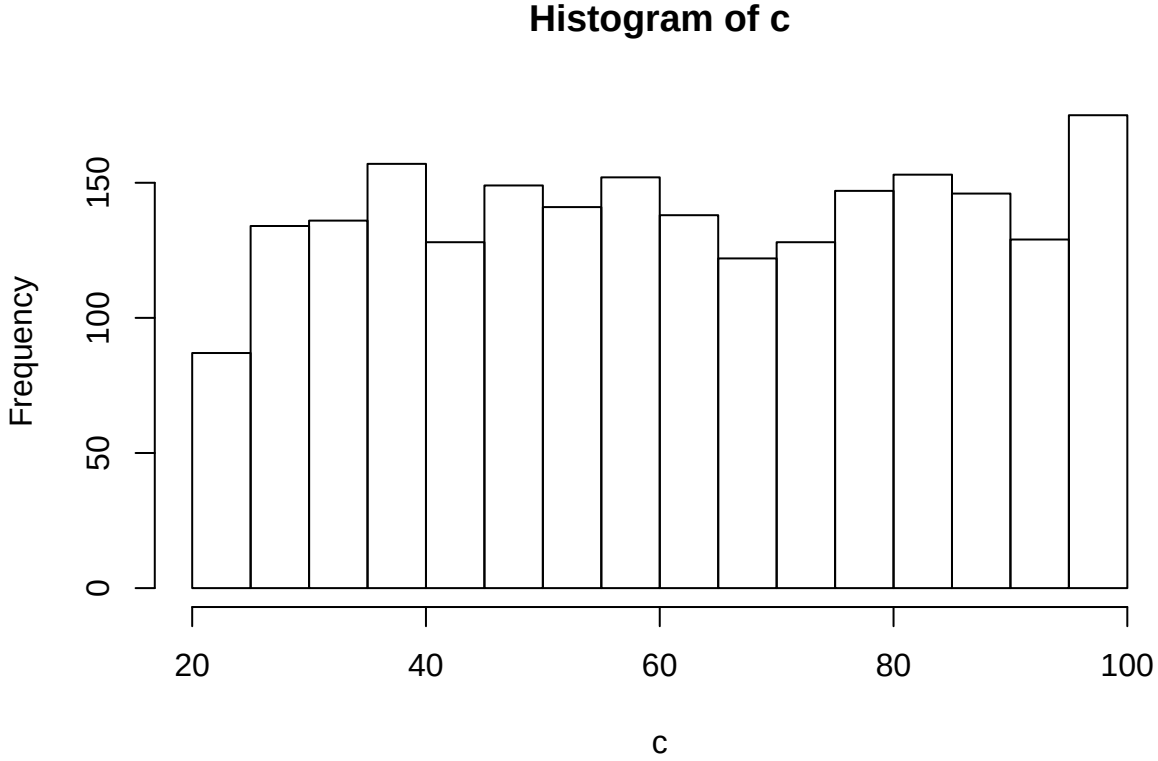
```
# Der Hashtag markiert Zeilen als Kommentar, alles was in der selben
# Zeile auf ihn folgt wird nicht mitinterpretiert
# runif verteitlt nun zehn Werte zufällig zwischen 20 und 200
(b <- runif(10, min = 20, max = 200))
#>  [1]  84.60762  70.67315 192.93810  87.19395  99.13927  93.28993  76.54687
#>  [8]  35.06041  68.92342 123.80406
(mean(b))
#> [1] 93.21768
b <- sort(b) # Überschreiben des alten b zu einem neuen sortierten b
```

Die bisher gezeigten Funktionen gehören zu den Paketen {base} und {stats} und sind, im Gebrauch von R.Studio immer vorhanden, anders als Pakete welche explizit geladen werden müssen. Auch zugänglich ist immer das Paket {graphics}:

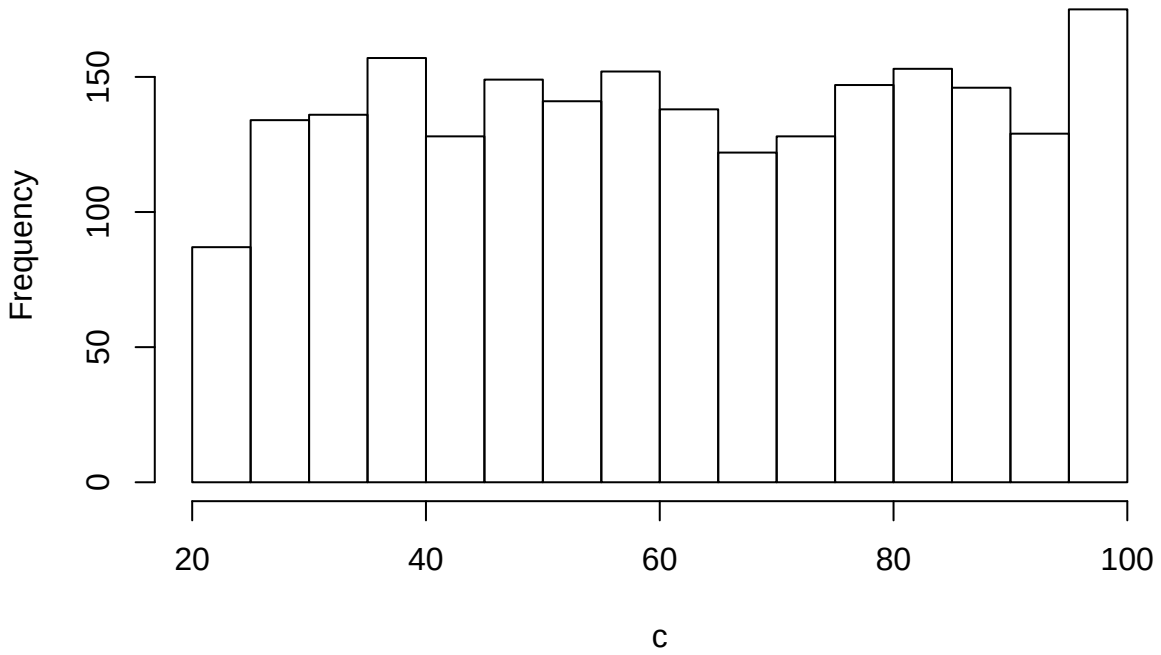
```
c <- runif(2222, 22, 100) # min und max kann man auch weglassen, ;-)
boxplot(b, c) # Ein einfacher Boxplot aus den Objekten a und b
```




```r
hist(c) # Histogramm
```




```r
nclass.Sturges(c) # Berechnung der Klassenbreite nach Sturges
#> [1] 13
nclass.scott(c) # Berechnung der Klassenbreite nach Scott
#> [1] 13
hist(c,
     nclass = 13, # Wählen der Klassenbreite
     main = "Klassenbreite = 13" # einfügen einer Überschrift
     )
```



Das Paket {graphics} enthält eine umfassenden Sammlung an einfachen Plottingfunktionen, wer sich dafür interessiert ist unter https://www.rdocumentation.org/packages/graphics/versions/3.6.2 an der richtigen Stelle.

Steuerungsparameter

Um beurteilen zu können ob ein Experiment Informationen liefert, die im Kontext zur Durchführung sinnvoll scheinen, können wir untersuchen wie die einzelnen Messwerte mit einander korrelieren. Im Fall einer vierfach-Bestimmung ist das einfach nachvollziehbar:

```
mean( c(1,2,3,4) )
#> [1] 2.5
mean( c(2,2.5,3,2.5))
#> [1] 2.5
```

Die technischen Replikate der ersten Messung liegen weit auseinander aber haben das selbe arithmetische Mittel wie die der zweiten Messung welche aber doch recht ähnliche Ergebnisse liefert!

Um einen Wert für die Abweichung der Messpunkte einer Messung zu einander zu können wir die Summe der Abweichungsquadrate SQ ermitteln:

$$SQ := \sum_{i=1}^{n} (x_i - \overline{x})^2 = (n-1)s_x^2$$

Wobei nun immer nur ein einzelner Punkt betrachtet wird, für die Messung selbst ist die Varianz s_x^2 von Interesse, sie beschreibt die Summe aller SQ:

$$s_x^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \overline{x})^2$$

Die dazugehörige Funktion in R heißt *var()*:

```r
var( c(1, 2, 3, 4))
#> [1] 1.666667
var( c(2,2.5,3,2.5))
#> [1] 0.1666667
var( rep(2.5, 4)) # rep() lässt einen Vektor aus vier mal 2,5 entstehen!
#> [1] 0
```

Die Standardabweichung $s = \sqrt{s_x^2}$ wiederum ist die positive Wurzel der Varianz und bekommt die den Namen *sd()*:

```r
sqrt(var(c(1, 2, 3, 4))) # sqrt() ist die Quadratwurzel
#> [1] 1.290994
var(c(1, 2, 3, 4)) ^ (1/2) # Wobei es auch so geht
#> [1] 1.290994
sd( c(1, 2, 3, 4))
#> [1] 1.290994
sd( c(2,2.5,3,2.5))
#> [1] 0.4082483
sd( rep(2.5, 4))
#> [1] 0
```

> R eignet sich im Labor super als Taschenrechner-Ersatz, Alle mathematischen Funktionen sind vorhanden, die Rechnungen können auch leicht modifiziert werden und wenn man sie als Skript schreibt kann super nachvollzogen werden was gerechnet wurde! Dies hilft auch sich an die Sprache R zu gewöhnen, sollte aber kein allzu aufwendiger Prozess werden :-) (Ich hab's ja auch geschafft)

Möchten wir etwas mehr über unseren Mittelwert erfahren dividieren wir die Standard-Abweichung s durch die Wurzel des Stichprobenumfang, also den Standardfehler $s_{\bar{x}} = \frac{s}{\sqrt{n}}$. Wir haben nun alle Werkzeuge um die Funktion für den Standardfehler selbst zu definieren. Dazu verwenden wir den Befehl *function()*:

```r
sde <- function(x) { # die Runde Klammer definiert das Argument der Funktion
  # Die geschweifte Klammer enthält die Definition der Funktion
  sd(x)/
    sqrt(length(x)) # length() ist die länge des Vektors von x
}
sde(c(1, 2, 3, 4))
#> [1] 0.6454972
sde(c(2,2.5,3,2.5))
#> [1] 0.2041241
sde(rep(2.5, 4))
#> [1] 0
```

Der Variationskoeffizient drückt prozentual die Abweichung der Standardabweichung zum arithmetischen Mittel aus $\left(cv = \frac{s}{\bar{x}} \cdot 100\%\right)$.

```r
cv <- function(x){
return( (sd(x)/mean(x))*100 ) # return() lässt, wie die doppelte Umklammerung das Ergebnis sofort ersch
}
cv(c(1, 2, 3, 4))
#> [1] 51.63978
cv(c(2,2.5,3,2.5))
#> [1] 16.32993
cv(rep(2.5, 4))
#> [1] 0
```

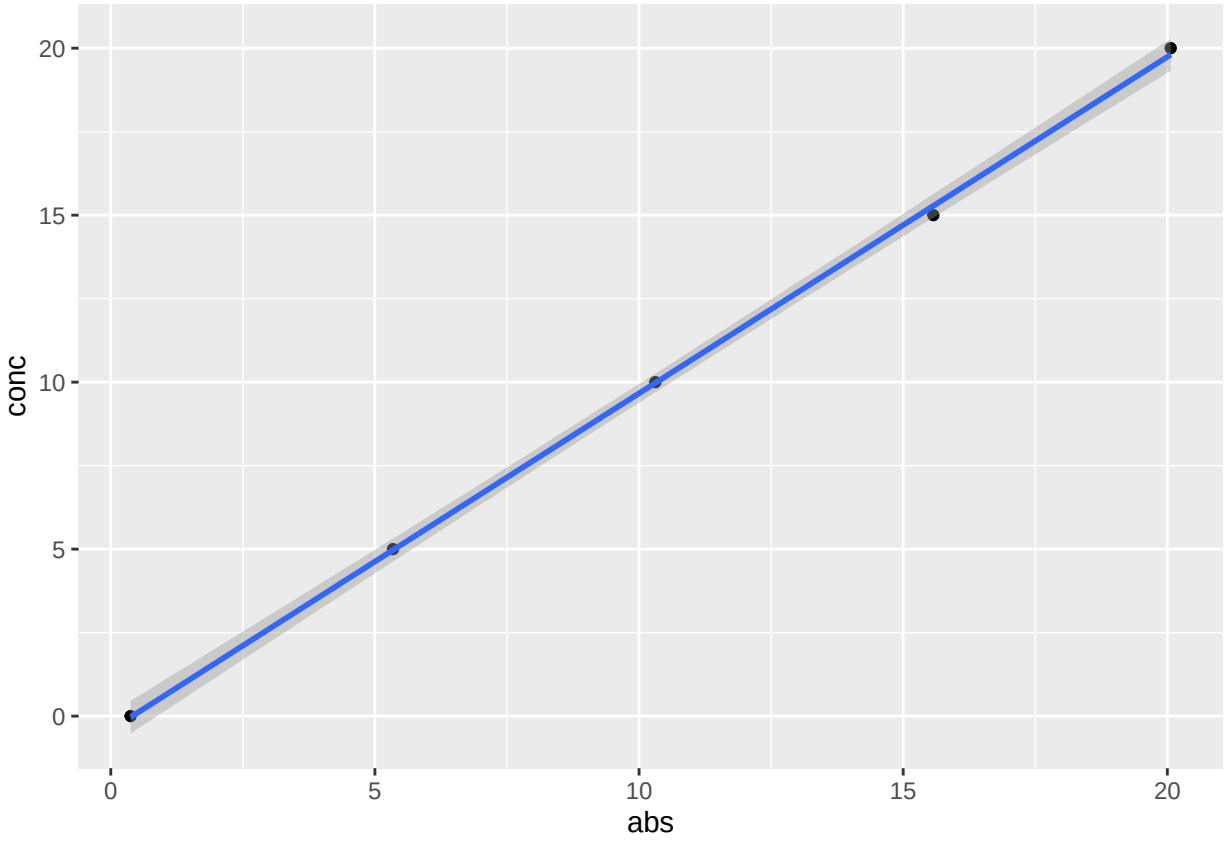
Der Variationskoeffizient lässt zu, dass wir verschiedene Stichproben mit unterschiedlichen Mitteln vergleichen:

```r
a <- c(500, 350, 400, 330, 370)
b <- c(50, 35, 40, 33, 37)
c <- c(234, 3, 44, 577, 9)
cv(a)
#> [1] 17.1047
cv(b)
#> [1] 17.1047
cv(c)
#> [1] 141.0778
```

Lineare Regression

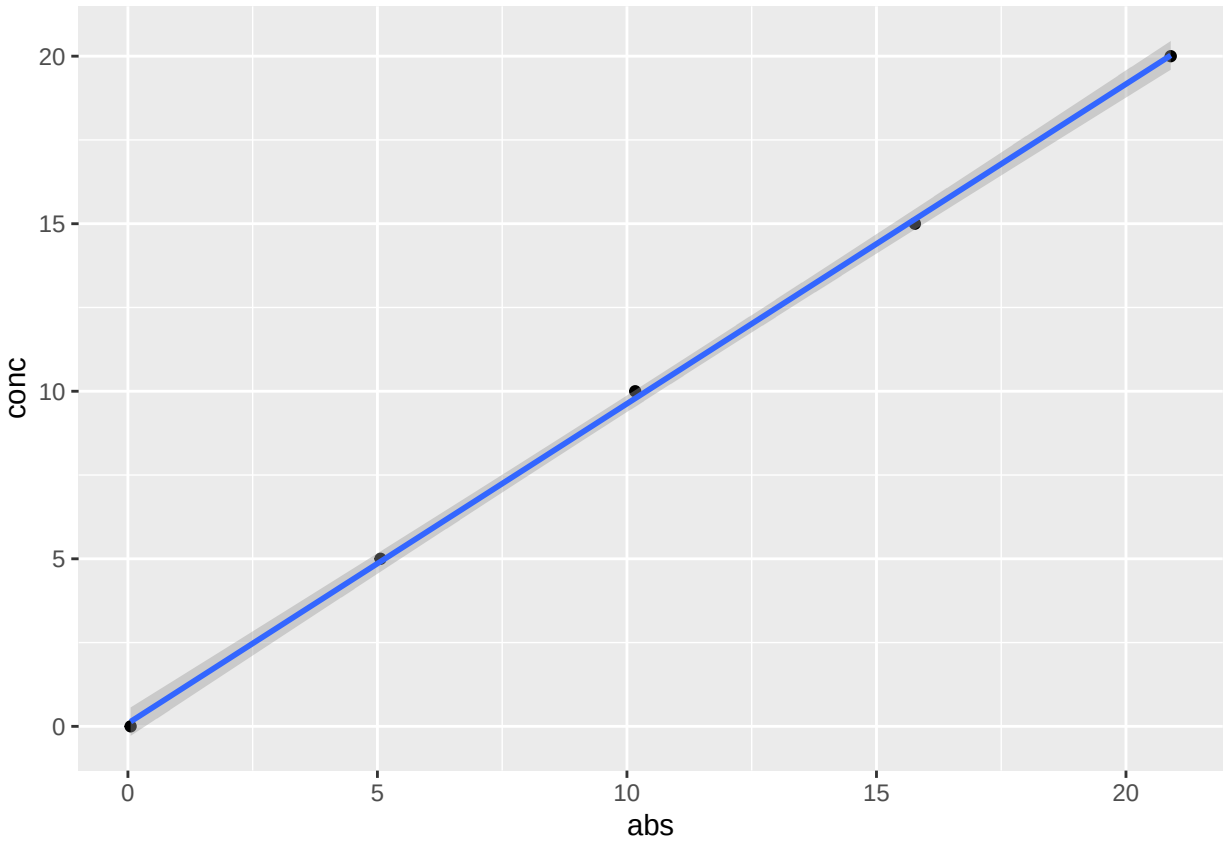
Die lineare Regression ist eine recht häufig verwendete Methode in der BT, als klassisches Beispiel ist die Konzentrationsbestimmung über eine Regrssion ("Eichgerade", "Kalibrationsgerade" etc.) zu nennen. Das Paket OlFaBA enthält die Funktion *plot_regression()* um schnell eine solche Regressionsgerade zu plotten:

```r
conc <- seq(0, 20, 5)
abs <- runif(5, 0, 1) + conc # Simmulierte Absorptionsmessung
plot_regression(abs, conc)  # Plotten den Absorption gegen Konzentration
```



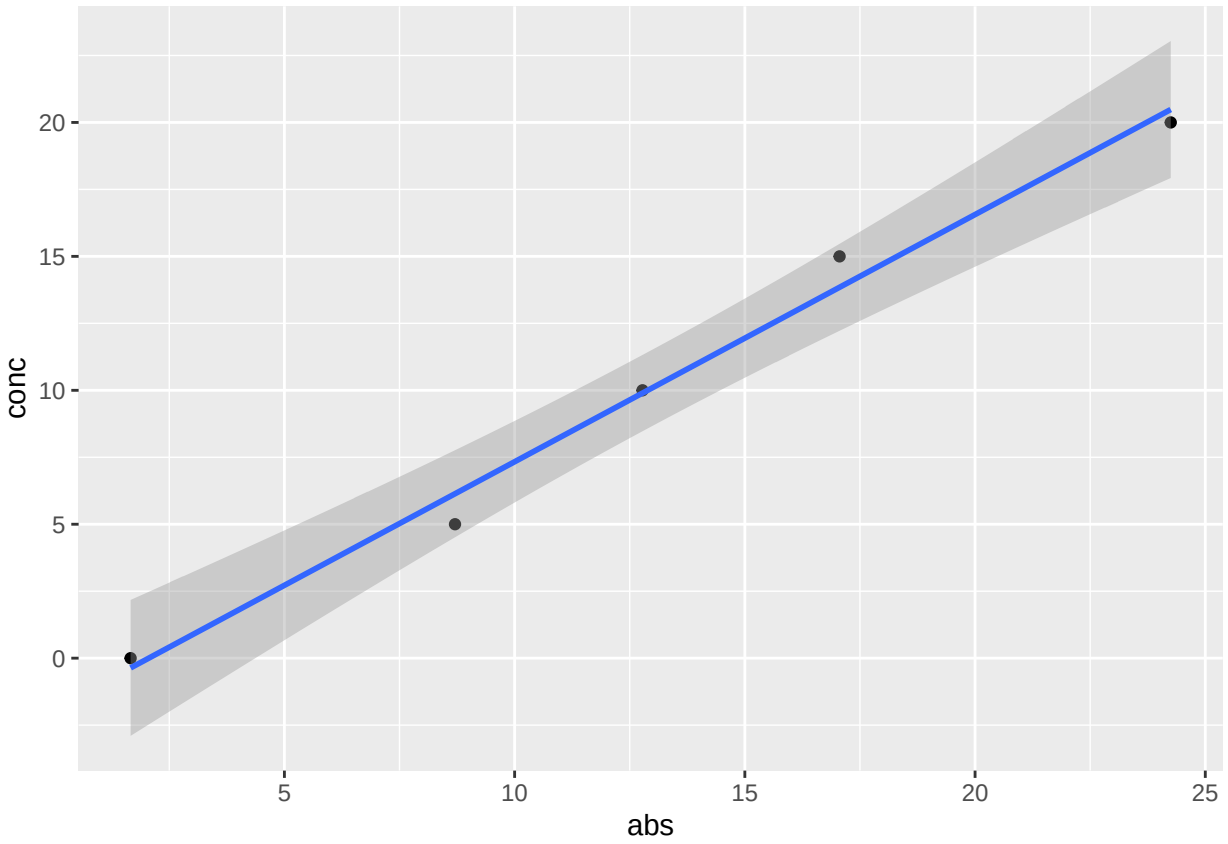
Dieser Plott zeigt uns, in grau hinterlegt die gleitenden durchnittliche Abweichung, auch ein Maß für die Zuverlässigkeit unserer Arbeitsweise, es könnt auch anders aussehen:

```r
conc <- seq(0, 20, 5)
abs <- sort(runif(5, 0, 1)) + conc # Simmulierte Absorptionsmessung
plot_regression(abs, conc)  # Plotten den Absorption gegen Konzentration
```



Aber schlimmer geht ja bekanntlich immer:

```
conc <- seq(0, 20, 5)
abs <- runif(5, 0, 5) + conc # Simmulierte Absorptionsmessung
plot_regression(abs, conc)  # Plotten den Absorption gegen Konzentration
```




```
Modell <- LinMod_CEv(abs, conc)
summary(Modell)
#>
#> Call:
#> stats::lm(formula = conc ~ abs)
#>
#> Residuals:
#>       1       2       3       4       5
#>  0.3631 -1.1384  0.1039  1.1541 -0.4827
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -1.89489    0.87962  -2.154 0.120229
#> abs          0.92279    0.05875  15.707 0.000561 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 1.001 on 3 degrees of freedom
#> Multiple R-squared:  0.988,  Adjusted R-squared:  0.984
#> F-statistic: 246.7 on 1 and 3 DF,  p-value: 0.0005609
```

Das R^2 gibt uns bekanntlich die Genauigkeit unserer Regressionsgerade an, es wird als Bestimmtheitsmaß bezeichnet. Alternativ wird im Laborjargon häufig auch der Begriff "Quadratfehler" verwendet. Um an diese Information zu kommen verwenden wir hierbei explizit eine Funktion aus dem Paket {base}, nämlich *summary()*, diese Funktion liefert uns eine Menge an Information, die wirklich relevanten sind hier zusammengestellt:

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-1.895	0.8796	-2.154	0.1202
abs	0.9228	0.05875	15.71	0.0005609

Table 2: Fitting linear model: conc ~ abs

Observations	Residual Std. Error	R^2	Adjusted R^2
5	1.001	0.988	0.984

Intercept ist der Schnittpunkt mit der y-Achse also b und der Steigung m als **conc** für $y = mx + b$. Damit kann jetzt auch weitergearbeitet werden, nun kann die Konzentration einer unbekannten Probe über ihre Absorption bestimmt werden. Dafür steht in {OlFaBA} die Funktion *conc_eval()* zur Verfügung, bevor wir diese jedoch verwenden muss *LinMod_CEv()* für die Standardreihe durchgeführt worden sein, im Beispiel mit realen Messdaten:

```
conc <- seq(0, 100, length.out = 6) # Ein Vektor der Länge 6, die Konzentrationen den Standards
abs <- c(.374, .5, .617, .78, .874, .985) # Die Real gemessenen Konzentrationen
conc_eval(abs_P = abs, abs_std = abs, conc_std = conc) #Berechnung der Konzentrationen der gemessenen A
#> [1] "\n-----------------------------------------------------------------\n                Estimate
#> attr(,"class")
#> [1] "knit_asis"
#> attr(,"knit_cacheable")
#> [1] NA
#> [1] -0.4795571 19.7550904 38.5444059 64.7209737 79.8166631 97.6424240
```

Teil der Funktion conc_eval() ist es, die errechneten Daten in einer Tabelle darzustellen, diese kann besonders gut bei der Verwendung von R Markdown zur Geltung kommen.

```
#> [1] "\n-----------------------------------------------------------------\n                Estimate
#> attr(,"class")
#> [1] "knit_asis"
#> attr(,"knit_cacheable")
#> [1] NA
#> [1] -0.4795571 19.7550904 38.5444059 64.7209737 79.8166631 97.6424240
```

Absorption	Ist-Konzentration	Berechnete Konzentration
0.374	0	-0.4796
0.5	20	19.76
0.617	40	38.54
0.78	60	64.72
0.874	80	79.82
0.985	100	97.64

Nicht-lineare Regressionen

Nicht-lineare Regression für die Michaelis-Menten-Funktion

Die Michaelis-Menten Gleichung kann mathematisch beschrieben werden als:

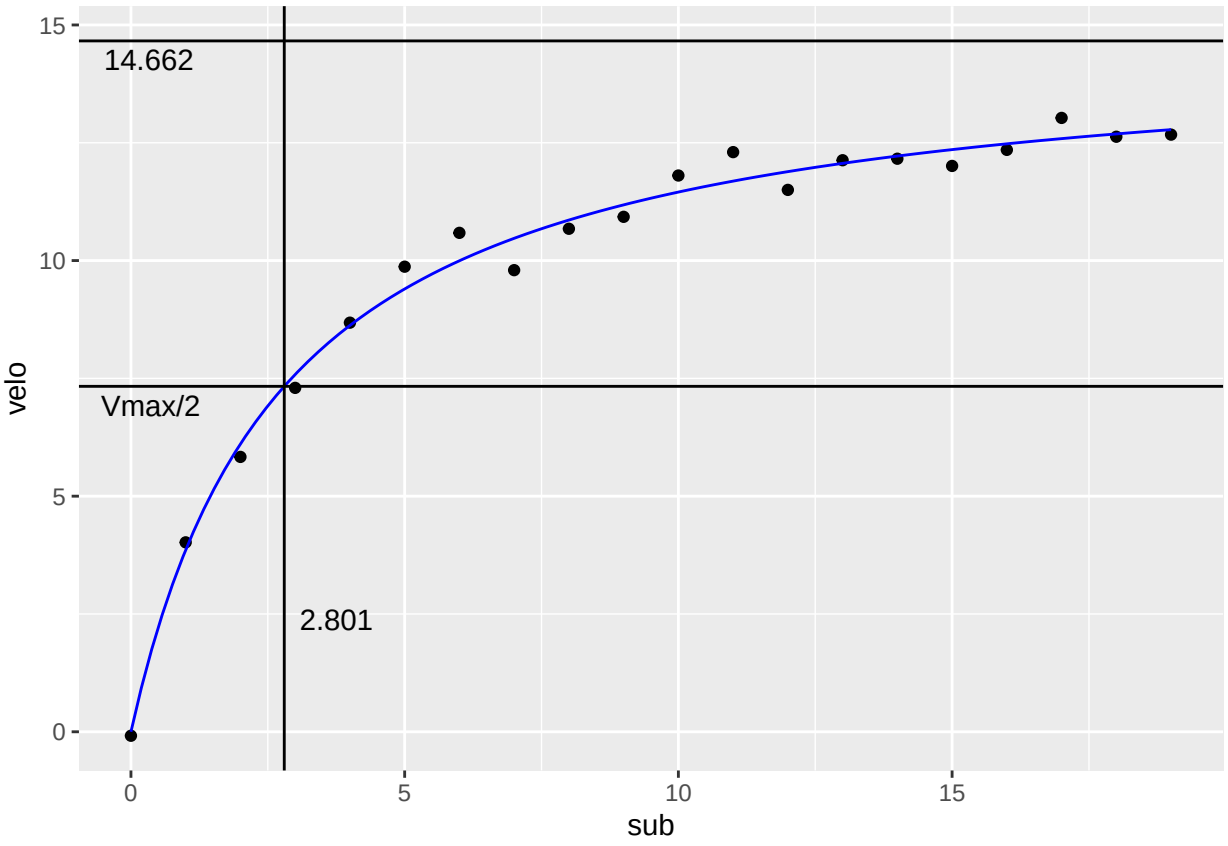
$$v_0 = \frac{v_{\max} \cdot [\mathrm{S}]}{K_\mathrm{m} + [\mathrm{S}]}$$

Wir können uns als nächstes wieder Test-Daten simulieren:

```
sub <-seq(0,19,1)
velo <-(
  (runif(1,14.7,15)*sub
  )/( # Könnt ihr die MM-Kinetik wiedererkennen? ;-)
    runif(1,2.5,3)+sub))+rnorm(20,0,.3)
Daten <- tibble(sub, velo)

fit <- nls(velo ~ (V_max * sub) / (K_m + sub),
           start = c(V_max = 15 , K_m = .5))
coef(fit)
#>    V_max      K_m
#> 14.66198  2.80069

ggplot(mapping = aes(x = sub, y = velo))+
  geom_point()+
  stat_function(fun = function(sub){ (coef(fit)[[1]] * sub) / ( coef(fit)[[2]] + sub)}, color = "blue")+
  geom_hline(yintercept = coef(fit)[[1]])+ # Vmax aus coefficents eintragen
  geom_text(aes(0,coef(fit)[[1]],label = round(coef(fit)[[1]], digits = 3), vjust = 1.4), hjust = .3 )+
  geom_hline(yintercept =( coef(fit)[[1]])/2,)+ # Vmax/2 aus coefficents
  geom_text(aes(0,(coef(fit)[[1]])/2,label = "Vmax/2", vjust = 1.4), hjust = .3)+ # Vmax/2 aus coeffice
  geom_vline(xintercept = coef(fit)[[2]])+ # Km aus coefficents
  geom_text(aes(0,coef(fit)[[2]],label = round(coef(fit)[[2]], digits = 3), vjust = 1.41), hjust = -2.3
```



Das hier funktioniert leider überhaupt nicht wie gedacht. . . : Die Start-Werte können nicht, wie im Beispiel oben, in die Funktion übertragen werden, ich würde mich sehr über Hilfe freuen, in non_linear_MM.R habe ich zusätzliche Kommentare eingefügt

Im nächsten Schritt soll ein Fitting für die Substratvariation gegen die Enzymaktivität vorgenommen werden:

```
# non_linear_MM(sub = sub, velo = velo, V_max_start = 10, K_m_start = 5)
```

Scott, David W. 1979. "On optimal and data-based histograms." *Biometrika* 66 (3): 605–10. https://doi.org/10.1093/biomet/66.3.605.

Sturges, Herbert A. 1926. "The Choice of a Class Interval." *Journal of the American Statistical Association* 21 (153). Taylor & Francis: 65–66. https://doi.org/10.1080/01621459.1926.10502161.