```
library(pander)
library(magrittr)
```

Auswertung von 96-Well Platten

Bei der Auswertung von 96-Well Platten (96WP) verfügt der Experimentator über eine größere Menge an Werten als im Normalfall. Dieses Plus kann aber in der Auswertung zu Schwierigkeiten führen, es absehbar, dass die Menge an Messdaten falsch importiert wird oder falsch weiterverarbeitet. Das Pipettierschema ist hierbei der Schlüssel zum Erfolg, wenn dieses korrekt übertragen wird können Fehler beim Datenimport oder bei der Durchführung erkannt werden. Die Messdaten werden in der Regel als Excel-Tabelle weitergegeben. Es ist zwingend notwendig den Datensatz als isolierte Tabelle ohne Überhang und mit sinnvollen Spaltennamen zu importieren. Beispiel wird die Konzentration an Humanserumalbumin (HSA) in verschiedenen Proben untersucht, im Datensatz sind sowohl die Messwerte einer Standardreihe, jene der Studentenurinproben als auch Patientenurinproben vorhanden. Das Orginalpipettierschema, in hellgrün die Standardreihe, in gelb und hellgelb die Studentenurinproben und die tieferen grüntöne markieren die Auftragungsstellen des Patientenurin:

	1	2	3	4	5	6	7	8	9	10	11	12
A		0	0	0								
B		5	5	5								
C		10	10	10								
D		15	15	15								
E		20	20	20								
F		30	30	30								
G		40	40	40								
H		50	50	50								

Der Import kann über readxl erfolgen:

```
library(readxl)
HSA <- read_excel("HSA_Arno.xlsx", sheet = "Tabelle2")
```

```
## New names:
## * `` -> ...1
```

Wir wissen, dass die Standards in aufsteigender Reihenfolge in den Spalten s2, s3 und s4 aufgetragen wurden. Unter Verwendung von dplyr::mutate() fällt es leicht den Durchschnitt zu

berechnen und gleich eine Spalte mit der gemittelten Konzentration einzuführen:

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------ tidyverse_conflicts() --
## x tidyr::extract()   masks magrittr::extract()
## x dplyr::filter()    masks stats::filter()
## x dplyr::lag()       masks stats::lag()
## x purrr::set_names() masks magrittr::set_names()
```

```
HSA <- mutate(HSA, Std=((HSA$s2 + HSA$s3 + HSA$s4) / 3))
pander(HSA)
```

Table 1: Table continues below

...1	s1	s2	s3	s4	s5	s6	s7	s8	s9
A	0.045	0.871	0.843	0.904	0.627	0.903	1.042	0.917	0.571
B	0.051	1.205	1.205	1.186	0.692	0.728	0.772	0.245	0.171
C	0.047	1.406	1.459	1.377	1.003	0.987	1.078	1.16	0.559
D	0.046	1.549	1.538	1.552	1.731	1.674	0.533	0.53	0.858
E	0.046	1.718	1.673	1.689	0.897	0.887	0.536	0.526	0.626
F	0.045	1.869	1.87	1.826	0.633	0.63	0.75	0.653	0.625
G	0.046	2.075	2.135	2.092	0.634	0.6	0.592	0.658	0.613
H	0.044	2.227	2.194	2.142	0.915	0.92	0.796	0.663	0.742

s10	s11	s12	Std
0.507	0.697	0.638	0.8727
0.49	0.635	0.686	1.199
0.562	0.055	0.201	1.414
0.677	0.935	0.972	1.546
0.516	0.955	0.995	1.693
0.477	0.394	0.43	1.855
0.621	0.934	0.848	2.101
0.707	1.086	0.355	2.188

Aus dem Standard kann über biotech::plot_regression() eine Regressionsgerade geplottete werden:

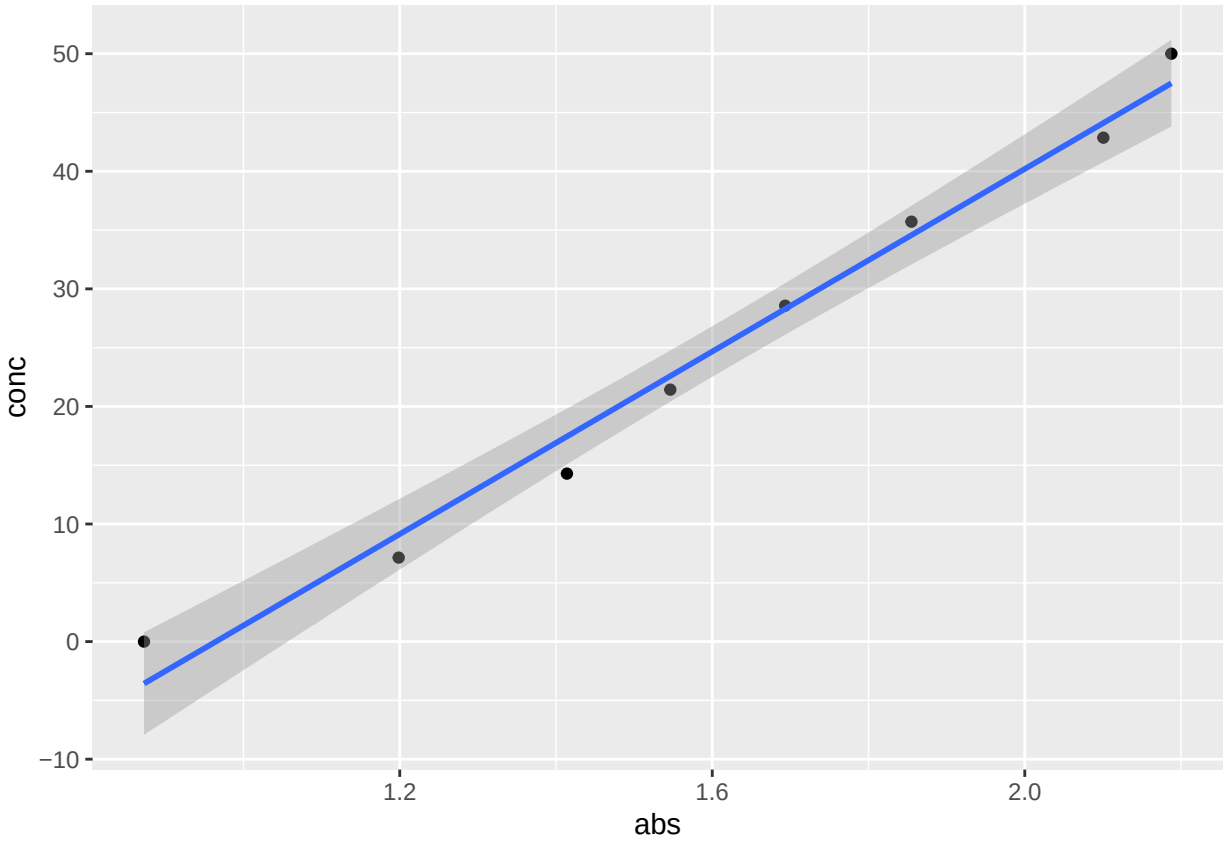
```
# Laden der Pakete
library(devtools)
```

```
## Loading required package: usethis
```

```
load_all("../../Biotech")
```

```
## Loading Biotech
```

```r
# Substratkonzentration errechnen:
subs <- seq(0, 50, length.out = length(HSA$Std))
# plotten des Modells
plot_regression(abs = HSA$Std, conc = subs)
```




```r
lm(subs[2:8]~HSA$Std[2:8]) %>%
    summary()
```

```
##
## Call:
## lm(formula = subs[2:8] ~ HSA$Std[2:8])
##
## Residuals:
##         1          2          3          4          5          6
##  0.5570020 -1.4928143  0.0006754  0.8680388  1.1092758 -2.2354836
##         7
##  1.1933058
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -44.586      2.929  -15.22 2.22e-05 ***
## HSA$Std[2:8]    42.690      1.677   25.45 1.75e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.479 on 5 degrees of freedom
## Multiple R-squared:  0.9923, Adjusted R-squared:  0.9908
```

```
## F-statistic: 647.7 on 1 and 5 DF,  p-value: 1.749e-06
```

Die geplottete Regressions Als nächstes wollen wir die Probenkonzentrationen untersuche, davor müssen wir diese aber auch aus den Rohdaten extrahieren. Die Versuchsdurchführung sieht vor, dass die Proben (Urinproben der Studenten) in zwei verschiedenen Konzentrationen in den selben Spalten auftragen, diese müssen nun getrennt werden:

```
# Die Spalten 5&6, 7&8 sowie 9&10 führen neben einander die
# Probenabsorption
Proben.1 <- c((HSA$s5 + HSA$s6)/2 , (HSA$s7[1:4]+HSA$s8[1:4])/2)
Proben.2 <- c((HSA$s9 + HSA$s10)/2 , (HSA$s7[5:8]+HSA$s8[5:8])/2)
```

Nach diesem Schritt können wir nun die Konzentration der verdünnten Proben bestimmen, es bleibt zu beachten, dass die Standardreihe im ng-Bereich konzentriert ist. Die Konzentrationen werden mit Biotech::conc_eval ermittelt:

```
Ergebnisse <- tibble(
       conc.Proben.1 = conc_eval(abs_P = Proben.1, HSA$Std, subs),
       conc.Proben.2 = conc_eval(abs_P = Proben.2, HSA$Std, subs)
)
```

```
##
## Call:
## stats::lm(formula = conc_std ~ abs_std)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1606 -1.4239 -0.4384  1.4871  3.5725
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -37.456      3.498  -10.71 3.92e-05 ***
## abs_std       38.828      2.105   18.45 1.64e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.488 on 6 degrees of freedom
## Multiple R-squared:  0.9827, Adjusted R-squared:  0.9798
## F-statistic: 340.2 on 1 and 6 DF,  p-value: 1.637e-06
##
##  [1]  -7.7529821  -9.8885185   1.1774433  28.6482082  -2.8218342
##  [6] -12.9365115 -13.4995166  -1.8317218   0.5756103 -17.7123477
## [11]   5.9921074 -16.8193052
##
## Call:
## stats::lm(formula = conc_std ~ abs_std)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1606 -1.4239 -0.4384  1.4871  3.5725
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -37.456      3.498  -10.71 3.92e-05 ***
## abs_std       38.828      2.105   18.45 1.64e-06 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.488 on 6 degrees of freedom
## Multiple R-squared:  0.9827, Adjusted R-squared:  0.9798
## F-statistic: 340.2 on 1 and 6 DF,  p-value: 1.637e-06
##
##  [1] -16.528096 -24.623720 -15.693295  -7.655912 -15.285602 -16.062160
##  [7] -13.499517  -9.325513 -16.838719 -10.218556 -13.188893  -9.131374
```

```
pander(Ergebnisse)
```

conc.Proben.1	conc.Proben.2
-7.753	-16.53
-9.889	-24.62
1.177	-15.69
28.65	-7.656
-2.822	-15.29
-12.94	-16.06
-13.5	-13.5
-1.832	-9.326
0.5756	-16.84
-17.71	-10.22
5.992	-13.19
-16.82	-9.131

Uns ist bekannt, dass die Proben der Studenten für Proben.1 und Proben.2 1:1000 und 1:5000 verdünnt wurden, diese Information kann gleich ergänzt werden:

```
Ergebnisse <- mutate(Ergebnisse, ist.conc.Proben.1 = conc.Proben.1 * 1000)
Ergebnisse <- mutate(Ergebnisse, ist.conc.Proben.2 = conc.Proben.2 * 5000)
pander(Ergebnisse)
```

conc.Proben.1	conc.Proben.2	ist.conc.Proben.1	ist.conc.Proben.2
-7.753	-16.53	-7753	-82640
-9.889	-24.62	-9889	-123119
1.177	-15.69	1177	-78466
28.65	-7.656	28648	-38280
-2.822	-15.29	-2822	-76428
-12.94	-16.06	-12937	-80311
-13.5	-13.5	-13500	-67498
-1.832	-9.326	-1832	-46628
0.5756	-16.84	575.6	-84194
-17.71	-10.22	-17712	-51093
5.992	-13.19	5992	-65944
-16.82	-9.131	-16819	-45657

Die Ergebnisse sind nun also die Konzentration der Proben in Nanogramm HSA je ml, die negativen Werte sind wahrscheinlich aus Proben entstanden, welche kein HSA enthielten und bei der Messung durch den Messfehler der teilweise veralteten Plattenleser entstanden. Durch die Multiplikation mit dem Verdünnungsfaktor werden sie verstärkt sind aber für die Auswertung nicht relevant. Die Urinproben der “Patienten” werden auch im ersten Schritt separiert:

```
Ergebnisse.Patienten <- tibble(
Patient.1 = (sum(HSA[1:2, 11], HSA[1:2, 12]) / 4),
Patient.2 = (sum(HSA[4:5, 11], HSA[4:5, 12]) /4)
)
pander(Ergebnisse.Patienten)
```

Patient.1	Patient.2
0.5822	0.7708

Um die Auswertung zu erleichtern bietet es sich an eine Funktion einzuführen um die Konzentration aus dem selben Modell zu berechnen:

```
cHSA <- function (abs.P) {
    conc_eval( abs.P,
               abs_std = HSA$Std,
               conc_std = subs
    )
}
```

Mit dieser Funktion können wir dann weiterrechnen:

```
Ergebnisse.Patienten <- mutate(
                                Ergebnisse.Patienten,
                                cHSA(Ergebnisse.Patienten$Patient.1),
                                cHSA(Ergebnisse.Patienten$Patient.2)
)
```

```
##
## Call:
## stats::lm(formula = conc_std ~ abs_std)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1606 -1.4239 -0.4384  1.4871  3.5725
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -37.456      3.498  -10.71 3.92e-05 ***
## abs_std       38.828      2.105   18.45 1.64e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.488 on 6 degrees of freedom
## Multiple R-squared:  0.9827, Adjusted R-squared:  0.9798
## F-statistic: 340.2 on 1 and 6 DF,  p-value: 1.637e-06
##
##   abs_std
## -14.84879
##
## Call:
## stats::lm(formula = conc_std ~ abs_std)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -3.1606 -1.4239 -0.4384  1.4871  3.5725
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -37.456      3.498  -10.71 3.92e-05 ***
## abs_std       38.828      2.105   18.45 1.64e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.488 on 6 degrees of freedom
## Multiple R-squared:  0.9827, Adjusted R-squared:  0.9798
## F-statistic: 340.2 on 1 and 6 DF,  p-value: 1.637e-06
##
##   abs_std
## -7.529721
```

```
pander(Ergebnisse.Patienten)
```

Table 6: Table continues below

Patient.1	Patient.2	cHSA(Ergebnisse.Patienten$Patient.1)
0.5822	0.7708	-14.85

cHSA(Ergebnisse.Patienten$Patient.2)
-7.53

Eine genauere Betrachtung der Ergebnisse liefert Gewissheit, dass auch hier etwas schief gelaufen ist während der Messung, es scheint, die Meisten Messwerte liegen außerhalb der gemessenen Standardreihe, wahrscheinlich ein Grund, dass viele negative Werte produziert wurden.

Auswertung eines weiteren Datensatz

Dieser wird wie der 1. importiert :

```
library(readxl)
HSA <- read_excel("HSA.xlsx", sheet = "Tabelle2")
```

```
## New names:
## * `` -> ...1
```

Und wie der erste verarbeitet, dafür wurden die Schritte der vorangegangenen Auswertung in einer Funktion zusammengefasst:

```
HSA_Cleanup <- function(HSA) {
list(  HSA <- mutate(HSA, Std=((HSA$s2 + HSA$s3 + HSA$s4) / 3)),
Proben <- tibble(
    Proben.1 = c((HSA$s5 + HSA$s6)/2 , (HSA$s7[1:4]+HSA$s8[1:4])/2),
    Proben.2 = c((HSA$s9 + HSA$s10)/2 , (HSA$s7[5:8]+HSA$s8[5:8])/2)
  ),
```

```
Patienten <- tibble(
    Patient.1 = (sum(HSA[1:2, 11], HSA[1:2, 12]) / 4),
    Patient.2 = (sum(HSA[4:5, 11], HSA[4:5, 12]) /4)
  ))
}

HSA.Clara <- HSA_Cleanup(HSA)
pander(HSA.Clara)
```

Table 8: Table continues below

...1	s1	s2	s3	s4	s5	s6	s7	s8
A	-42.11	-4.005	-5.088	-7.019	4.85	4.426	-6.501	-5.983
B	-42.01	-1.462	0.1868	2.071	-8.15	-4.853	17.61	13.99
C	-41.97	12.95	10.27	10.08	-8.197	13.89	1.647	0.2339
D	-42.01	18.74	19.31	17.14	19.12	19.07	21.66	18.6
E	-41.68	28.12	29.06	29.86	8.712	4.708	-8.338	-10.79
F	-41.87	33.11	33.72	30.09	52.42	48.98	-10.93	-12.44
G	-42.01	41.3	42.91	36.45	14.83	7.629	-12.25	-10.98
H	-41.97	42.53	40.93	48.75	2.542	0.1397	-4.382	-7.773

s9	s10	s11	s12	Std
-6.69	-9.233	13.94	17.14	-5.371
3.295	4.755	19.45	15.31	0.2653
-10.65	-6.925	-21.71	-19.59	11.1
-8.715	-10.88	-4.759	-3.628	18.4
-9.469	-4.146	-2.121	0.04554	29.01
-5.559	-6.831	-24.26	-20.4	32.31
-4.994	-0.4726	13.33	9.701	40.22
3.672	-1.509	14.18	18.27	44.07

-

Proben.1	Proben.2
4.638	-7.961
-6.501	4.025
2.848	-8.786
19.1	-9.798
6.71	-6.807
50.7	-6.195
11.23	-2.733
1.341	1.082
-6.242	-9.563
15.8	-11.68
0.9404	-11.61
20.13	-6.077

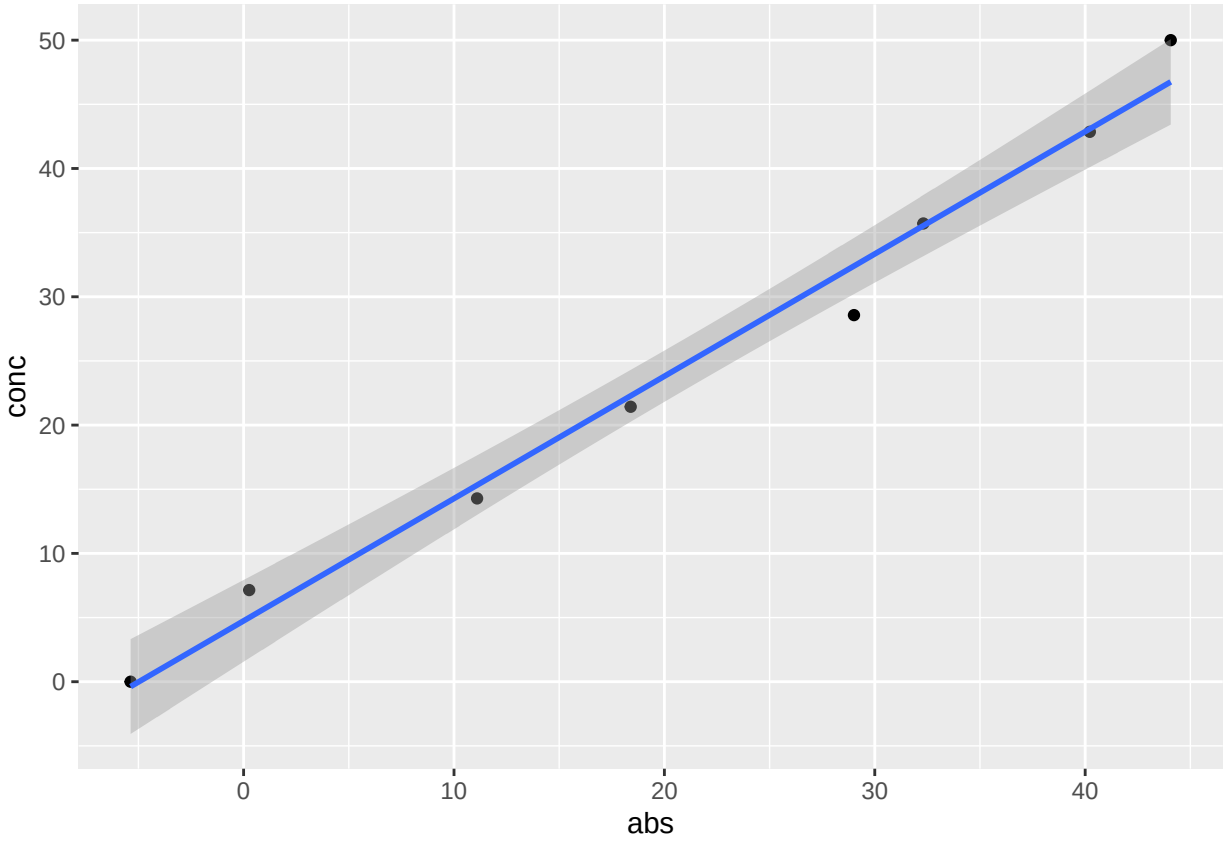
-

Patient.1	Patient.2
7.228	-5.477

-

Die Standardreihe kann geplottet werden:

```
plot_regression(abs = HSA.Clara[[1]]$Std[1:8], conc = subs[1:8] )
```



Der erste Wert der Standardreihe ist negativ, wir möchten nun untersuchen, in wie weit dieser und der 4. Wert das Bestimmtheitsmaß unserer Regression beeinflusst:

```
lm(HSA.Clara[[1]]$Std[1:8]~subs[1:8]) %>%
    summary()
```

```
##
## Call:
## lm(formula = HSA.Clara[[1]]$Std[1:8] ~ subs[1:8])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0196 -1.2201  0.2507  0.8600  4.0705
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.58719    1.54829  -2.963   0.0252 *
## subs[1:8]    1.03349    0.05182  19.945 1.03e-06 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.399 on 6 degrees of freedom
## Multiple R-squared:  0.9851, Adjusted R-squared:  0.9827
## F-statistic: 397.8 on 1 and 6 DF,  p-value: 1.031e-06
```

```
lm(HSA.Clara[[1]]$Std[2:8]~subs[2:8]) %>%
    summary()
```

```
##
## Call:
## lm(formula = HSA.Clara[[1]]$Std[2:8] ~ subs[2:8])
##
## Residuals:
##       1       2       3       4       5       6       7
## -2.97734  0.58538  0.61565  3.95858 -0.01458  0.62799 -2.79568
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.02742    2.18655  -1.842    0.125
## subs[2:8]    1.01781    0.06845  14.869 2.49e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.587 on 5 degrees of freedom
## Multiple R-squared:  0.9779, Adjusted R-squared:  0.9735
## F-statistic: 221.1 on 1 and 5 DF,  p-value: 2.489e-05
```

```
lm(HSA.Clara[[1]]$Std[-4]~subs[-4]) %>%
    summary()
```

```
##
## Call:
## lm(formula = HSA.Clara[[1]]$Std[-4] ~ subs[-4])
##
## Residuals:
##       1       2       3       4       5       6       7
## -0.62267 -2.38002  1.05925  4.18554  0.08893  0.60804 -2.93908
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.74821    1.73912   -2.73   0.0413 *
## subs[-4]     1.03510    0.05628   18.39 8.74e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.596 on 5 degrees of freedom
## Multiple R-squared:  0.9854, Adjusted R-squared:  0.9825
## F-statistic: 338.2 on 1 and 5 DF,  p-value: 8.742e-06
```

Wir sehen dass die Beeinflussung eher gering ist und gehen daher nicht weiter auf diese Umstände ein. Für die Konzentrationsberechnung definieren wir eine neue Funktion:

```
cHSA <- function(HSA, Proben.x, subs) {
  conc_eval(abs_P = Proben.x, abs_std = HSA[[1]]$Std[2:8], conc_std = subs[2:8])
}
```

Und wenden sie an:

```
Ergebnisse.Proben <- tibble(
    ist.conc.1 = 5000 * cHSA(HSA = HSA.Clara, Proben.x = HSA.Clara[[2]]$Proben.1, subs = s
    ist.conc.2 = 1000 * cHSA(HSA = HSA.Clara, Proben.x = HSA.Clara[[2]]$Proben.2, subs = s
    )
```

```
##
## Call:
## stats::lm(formula = conc_std ~ abs_std)
##
## Residuals:
##       1       2       3       4       5       6       7
##  2.3867 -0.8783 -0.7495 -3.8033  0.1720 -0.2874  3.1599
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.50126    1.87697   2.398   0.0618 .
## abs_std      0.96077    0.06461  14.869 2.49e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.514 on 5 degrees of freedom
## Multiple R-squared:  0.9779, Adjusted R-squared:  0.9735
## F-statistic: 221.1 on 1 and 5 DF,  p-value: 2.489e-05
##
##  [1]  8.957035 -1.744957  7.237476 22.849262 10.948103 53.213054 15.292252
##  [8]  5.789426 -1.496074 19.681653  5.404788 23.844796
##
## Call:
## stats::lm(formula = conc_std ~ abs_std)
##
## Residuals:
##       1       2       3       4       5       6       7
##  2.3867 -0.8783 -0.7495 -3.8033  0.1720 -0.2874  3.1599
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.50126    1.87697   2.398   0.0618 .
## abs_std      0.96077    0.06461  14.869 2.49e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.514 on 5 degrees of freedom
## Multiple R-squared:  0.9779, Adjusted R-squared:  0.9735
## F-statistic: 221.1 on 1 and 5 DF,  p-value: 2.489e-05
##
##  [1] -3.147756  8.368765 -3.939658 -4.912566 -2.039092 -1.450822  1.875167
##  [8]  5.540543 -4.686308 -6.722628 -6.654751 -1.337693
```

```
pander(Ergebnisse.Proben)
```

ist.conc.1	ist.conc.2
44785	-3148
-8725	8369

ist.conc.1	ist.conc.2
36187	-3940
114246	-4913
54741	-2039
266065	-1451
76461	1875
28947	5541
-7480	-4686
98408	-6723
27024	-6655
119224	-1338

Das selbe noch für die Patientenproben:

```
Ergebnisse.Patienten <- tibble(
    ist.conc.P1 =  cHSA(HSA = HSA.Clara, Proben.x = HSA.Clara[[3]]$Patient.1, subs = subs)
ist.conc.P2 =  cHSA(HSA = HSA.Clara, Proben.x = HSA.Clara[[3]]$Patient.2, subs = subs)
)
```

```
##
## Call:
## stats::lm(formula = conc_std ~ abs_std)
##
## Residuals:
##       1       2       3       4       5       6       7
##  2.3867 -0.8783 -0.7495 -3.8033  0.1720 -0.2874  3.1599
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.50126    1.87697   2.398   0.0618 .
## abs_std      0.96077    0.06461  14.869 2.49e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.514 on 5 degrees of freedom
## Multiple R-squared:  0.9779, Adjusted R-squared:  0.9735
## F-statistic: 221.1 on 1 and 5 DF,  p-value: 2.489e-05
##
##  abs_std
## 11.44587
##
## Call:
## stats::lm(formula = conc_std ~ abs_std)
##
## Residuals:
##       1       2       3       4       5       6       7
##  2.3867 -0.8783 -0.7495 -3.8033  0.1720 -0.2874  3.1599
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.50126    1.87697   2.398   0.0618 .
## abs_std      0.96077    0.06461  14.869 2.49e-05 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.514 on 5 degrees of freedom
## Multiple R-squared:  0.9779, Adjusted R-squared:  0.9735
## F-statistic: 221.1 on 1 and 5 DF,  p-value: 2.489e-05
##
##    abs_std
## -0.7607361
```

```
pander(Ergebnisse.Patienten)
```

ist.conc.P1	ist.conc.P2
11.45	-0.7607