*Corrélation LOC * PNS*

```
cor.test(df_participants$PNS, df_participants$LOC, exact = T, conf.level =
0.95)

        Pearson's product-moment correlation

data:  df_participants$PNS and df_participants$LOC
t = 3.0027, df = 190, p-value = 0.003036
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.07345525 0.34408660
sample estimates:
      cor
0.2128495

> cor.test(df_participants_cleaned$PNS, df_participants_cleaned$LOC, exact
= T, conf.level = 0.95)

        Pearson's product-moment correlation

data:  df_participants_cleaned$PNS and df_participants_cleaned$LOC
t = 2.8369, df = 136, p-value = 0.005253
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.07210961 0.38814035
sample estimates:
      cor
0.2363663
```

Régressions linéaires multiples

```
model.linear.outliers <- lm(FA_corr ~ loss_control+PNS+LOC+PNS*loss_control
+LOC*loss_control,
+                                       data = df_participants)
>
> model.linear.cleaned <- lm(FA_corr ~ loss_control+PNS+LOC+PNS*loss_contro
l+LOC*loss_control,
+                 data = df_participants_cleaned)
>
> export_summs(model.linear.outliers, model.linear.cleaned, stars=c("*"=0.1
, "**" = 0.05, "***" = 0.01), digits= 3, robust = "HC2",
+               model.names = c("Modèle 1 - OLS avec outliers", "Modèle 2 -
OLS sans outliers"))
```

	Modèle 1 - OLS avec outliers	Modèle 2 - OLS sans outliers
(Intercept)	0.220 *	0.055
	(0.113)	(0.100)
loss_control	-0.094	0.051
	(0.143)	(0.138)
PNS	-0.001	0.002
	(0.002)	(0.002)
LOC	0.003	0.003
	(0.003)	(0.003)
loss_control:PNS	-0.000	-0.004
	(0.003)	(0.003)
loss_control:LOC	0.004	0.004
	(0.004)	(0.004)
N	192	138
R2	0.038	0.041

Standard errors are heteroskedasticity robust. *** p < 0.01; ** p < 0.05; * p < 0.1.

Column names: names, Modèle 1 - OLS avec outliers, Modèle 2 - OLS sans outliers

Bêta régressions

```
model.beta.reg.outliers <- betareg(FA_corr ~ loss_control+PNS+LOC+PNS*loss_
control+LOC*loss_control, data = df_participants)
```

```
> model.beta.reg <- betareg(FA_corr ~ loss_control+PNS+LOC+PNS*loss_control
+LOC*loss_control, data = df_participants_cleaned)
>
> export_summs(model.beta.reg.outliers, model.beta.reg, stars=c("*"=0.1, "*
*" = 0.05, "***" = 0.01), digits= 3, robust = "HC2",
+              model.names = c("Modèle 1 - bêta rég avec outliers", "Modèle
2 - bêta reg sans outliers"))
```

	Modèle 1 - bêta rég avec outliers	Modèle 2 - bêta reg sans outliers
(Intercept)	-1.469 ***	-2.187 ***
	(0.528)	(0.646)
loss_control	-0.270	0.251
	(0.732)	(0.886)
PNS	0.000	0.013
	(0.008)	(0.010)
LOC	0.013	0.014
	(0.013)	(0.015)
loss_control:PNS	-0.003	-0.015
	(0.012)	(0.015)
loss_control:LOC	0.014	0.014
	(0.019)	(0.022)
(phi)	6.012 ***	6.283 ***
	(0.585)	(0.728)
nobs	192	138
pseudo.r.squared	0.028	0.044
df.null	190.000	136.000
logLik	90.615	75.915
AIC	-167.229	-137.831
BIC	-144.427	-117.340
df.residual	185.000	131.000
nobs.1	192.000	138.000

```
Standard errors are heteroskedasticity robust.  *** p < 0.01; ** p < 0.05; * p <
0.1.
Column names: names, Modèle 1 - bêta rég avec outliers, Modèle 2 - bêta reg sans outliers
```

<u>Convert coefficients of bêta régression (R) to compare to linear regression</u>

➪ coefficient par coefficient (alpha = intercept ; beta = coeff to convert)

```
convert_coeff_beta <- function(alpha, beta) {
  p0 =  1/(1+exp(-(alpha + beta*3)))
  p1 =  1/(1+exp(-(alpha + beta*4)))
  coeff_simple <- p1 - p0
  return(coeff_simple)
}
```

<u>Théorie de la detection du signal</u>

```
# to compute d' / c when Hits = 1 or FA = 0 (often when low nb of trials =
high sampling variability), we use loglinear transformation
# which seems to work quite well (Hautus, 1995; Stanislaw & Todorov, 1999)

      HIT_corr = (sum(signal == 1 & correct == 1) + 0.5)/ (nb_signal + 1),
     FA_corr = (sum(signal == 0 & correct == 0 & response != "timeout") +
0.5) / (nb_noise + 1),
      dprime = round(qnorm(HIT_corr) - qnorm(FA_corr),2),
    c = round(-0.5 * (qnorm(HIT_corr) + qnorm(FA_corr)), 2)
```

<u>Tests t d’ et c</u>

d'

```
> t.test(dprime ~ loss_control, data = df_participants)

	Welch Two Sample t-test

data:  dprime by loss_control
t = 0.60783, df = 189.75, p-value = 0.544
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.1534243  0.2900910
sample estimates:
mean in group 0 mean in group 1 
       1.169583        1.101250 
```

```
> t.test(dprime ~ loss_control, data = df_participants_cleaned)

	Welch Two Sample t-test

data:  dprime by loss_control
t = -0.113, df = 130.92, p-value = 0.9102
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.2479260  0.2211331
sample estimates:
mean in group 0 mean in group 1 
       1.290694        1.304091 
```

c (decision criterion)

```
t.test(c ~ loss_control, data = df_participants)

	Welch Two Sample t-test

data:  c by loss_control
t = 0.32558, df = 189.39, p-value = 0.7451
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.1022250  0.1426416
sample estimates:
mean in group 0 mean in group 1 
      0.1595833       0.1393750 
```

```
> t.test(c ~ loss_control, data = df_participants_cleaned)

	Welch Two Sample t-test

data:  c by loss_control
t = 0.0055037, df = 129.78, p-value = 0.9956
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.1538898  0.1547484
sample estimates:
mean in group 0 mean in group 1 
      0.1702778       0.1698485 
```