4.3 Exercise: random intercepts and slopes

Benjamin Rosenbaum

October 27, 2022

We will extend the last model by including random slopes, too.

Setup

```
rm(list=ls())
library(rstan)
library(coda)
library(BayesianTools)
library(brms)

setwd("~/Nextcloud/teaching Bayes 2021")

rstan_options(auto_write = TRUE)
options(mc.cores = 4)
```

Read dataset

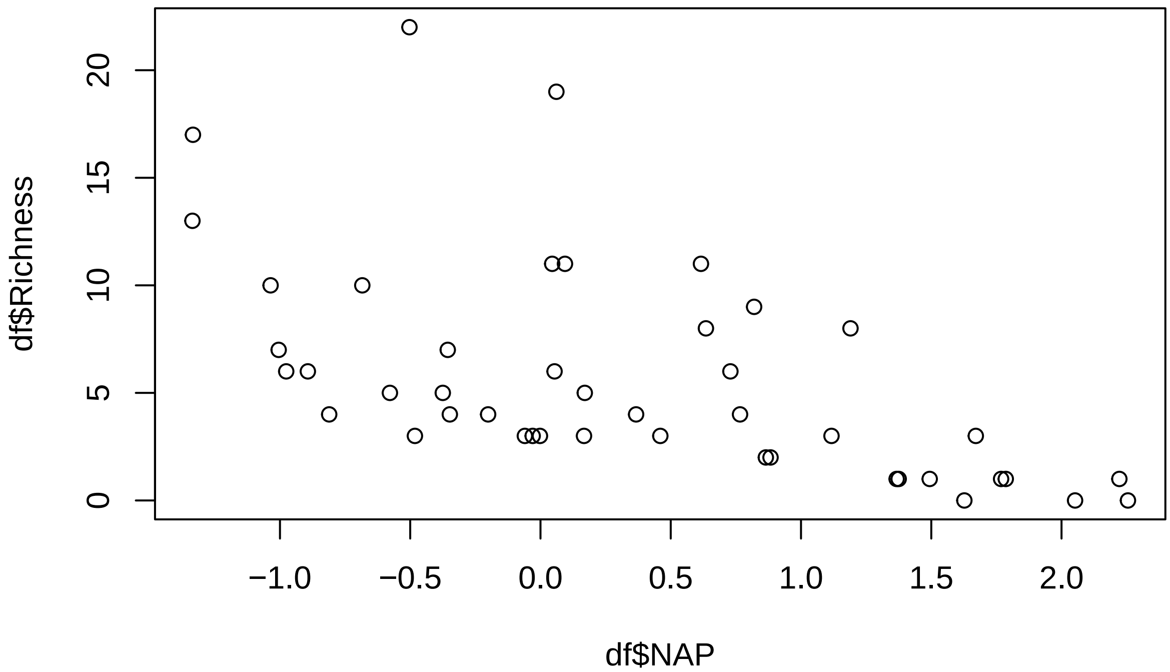
```
df = read.table("data/RIKZ.txt", header=TRUE)
head(df)
```

```
##   Sample Richness Exposure    NAP Beach
## 1      1       11       10  0.045     1
## 2      2       10       10 -1.036     1
## 3      3       13       10 -1.336     1
## 4      4       11       10  0.616     1
## 5      5       10       10 -0.684     1
## 6      6        8        8  1.190     2
```

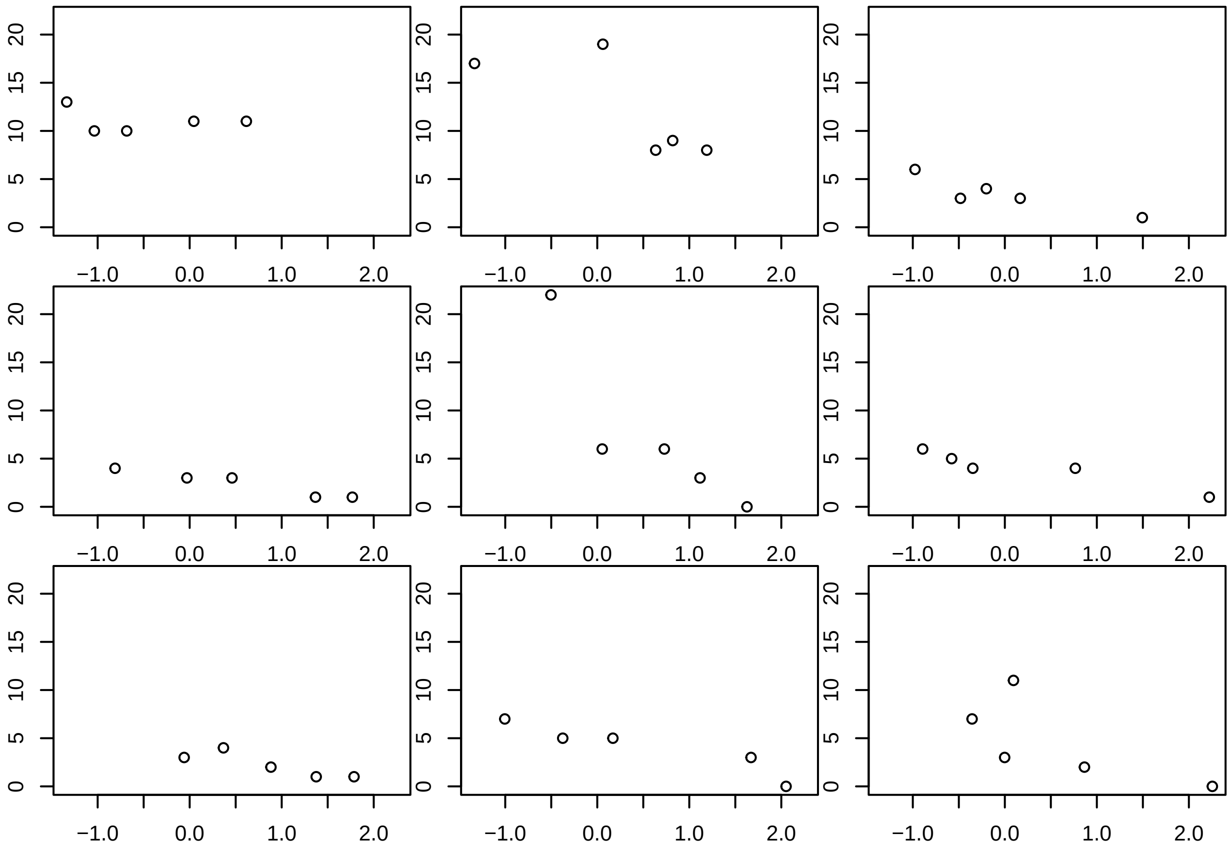
```
str(df)
```

```
## 'data.frame':    45 obs. of  5 variables:
##  $ Sample  : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Richness: int  11 10 13 11 10 8 9 8 19 17 ...
##  $ Exposure: int  10 10 10 10 10 8 8 8 8 8 ...
##  $ NAP     : num  0.045 -1.036 -1.336 0.616 -0.684 ...
##  $ Beach   : int  1 1 1 1 1 2 2 2 2 2 ...
```

```
par(mfrow=c(1,1))
plot(df$NAP, df$Richness)
```




```
par(mfrow=c(3,3), mar=c(1,1,1,1), oma=c(1,1,0,0))
for (i in 1:9){
  df.sub=subset(df, df$Beach==i)
  plot(df.sub$NAP,  df.sub$Richness,
       xlim=range(df$NAP),
       ylim=range(df$Richness)  )
}
```



Random intercepts model

We will fit linear regression lines to each group (`Beach`) as follows:

$$
\begin{aligned}
y_i &\sim \text{normal}(a_{group(i)} + b_{group(i)} \cdot x_i, \sigma) \quad i = 1, ..., n \quad (n \text{ observations}) \\
a_j &\sim \text{normal}(\mu_a, \sigma_a) \quad j = 1, ..., m \quad (m \text{ groups}) \\
b_j &\sim \text{normal}(\mu_b, \sigma_b) \quad j = 1, ..., m \quad (m \text{ groups})
\end{aligned}
$$

Here, a_j and b_j are group-level intercepts and slopes, which are allowed to vary (partial pooling).

Both have their own means (μ_a, μ_b) and standard deviations (σ_a, σ_b), which are also free parameters to be estimated.

So this is a random intercepts and slopes linear regression, lm-formulation would be `y ~ x + (x|group)`, which is short for `y ~ 1+x + (1+x|group)`.

Here, an intercept a_j and a slope b_j are independent. Usually they are modeled with a correlation (standard in `lme4` and also in `brms`), but we leave that out in Stan.

```
data = list(y = df$Richness,
            x = df$NAP,
            group = df$Beach,
            n = nrow(df),
            n_group = 9)

data
```

```
## $y
```

```
##  [1] 11 10 13 11 10  8  9  8 19 17  6  1  4  3  3  1  3  3  1  4  3 22  6  0  6
## [26]  5  4  1  6  4  2  1  1  3  4  3  5  7  5  0  7 11  3  0  2
##
## $x
##  [1]  0.045 -1.036 -1.336  0.616 -0.684  1.190  0.820  0.635  0.061 -1.334
## [11] -0.976  1.494 -0.201 -0.482  0.167  1.768 -0.030  0.460  1.367 -0.811
## [21]  1.117 -0.503  0.729  1.627  0.054 -0.578 -0.348  2.222 -0.893  0.766
## [31]  0.883  1.786  1.375 -0.060  0.367  1.671 -0.375 -1.005  0.170  2.052
## [41] -0.356  0.094 -0.002  2.255  0.865
##
## $group
##  [1] 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5 6 6 6 6 6 7 7 7 7 7 8 8 8
## [39] 8 8 9 9 9 9 9
##
## $n
## [1] 45
##
## $n_group
## [1] 9
```

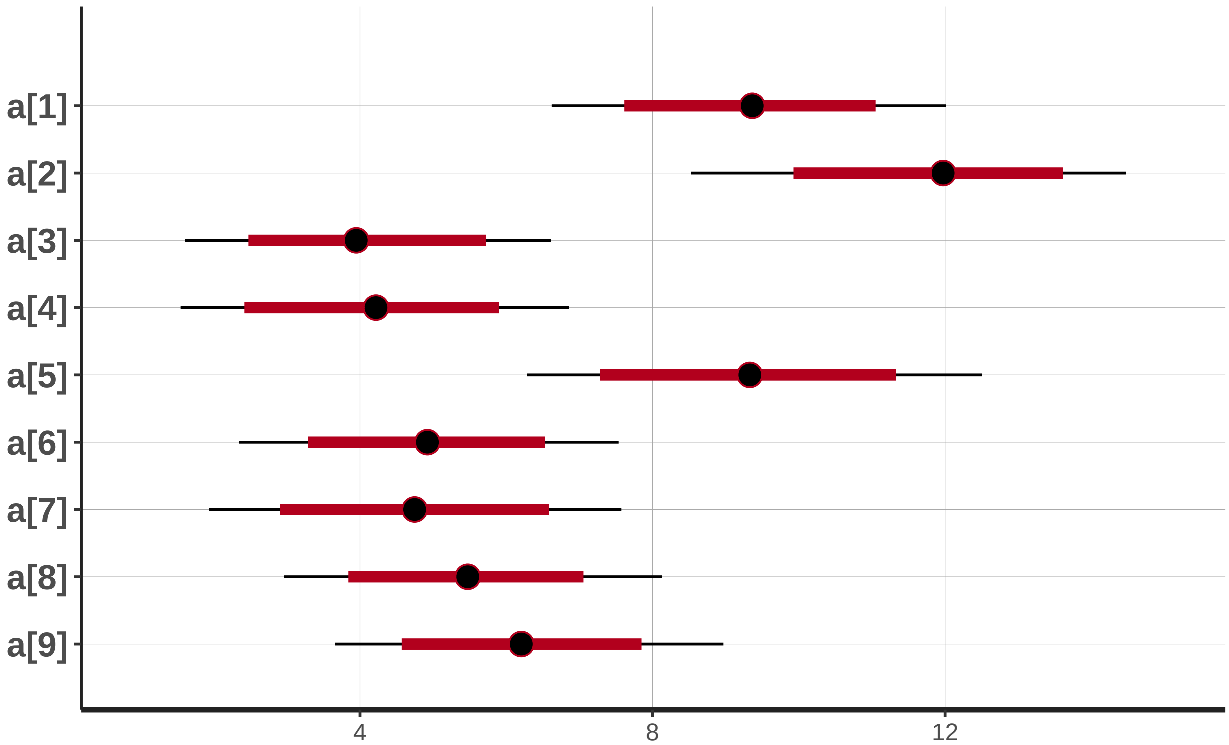
```
stan_code_partpool = '
data {
  int n;
  int n_group;
  real y[n];
  real x[n];
  int group[n];
}
parameters {
  real a[n_group];
  real b[n_group];
  real<lower=0> sigma;
  real mu_a;
  real<lower=0> sigma_a;
  real mu_b;
  real<lower=0> sigma_b;
}
model {
  // priors
  mu_a ~ normal(0,10);
  mu_b ~ normal(0,10);
  sigma_a ~ cauchy(0,1);
  sigma_b ~ cauchy(0,1);
  for (j in 1:n_group){
    a[j] ~ normal(mu_a,sigma_a);
    b[j] ~ normal(mu_b,sigma_b);
  }
  sigma ~ normal(0,10);
  // likelihood
  for(i in 1:n){
    y[i] ~ normal( a[ group[i] ] + b[ group[i] ] *x[i] , sigma);
  }
}
'
```

```r
stan_model_partpool = stan_model(model_code=stan_code_partpool)
fit_partpool  = sampling(stan_model_partpool, data=data)
```

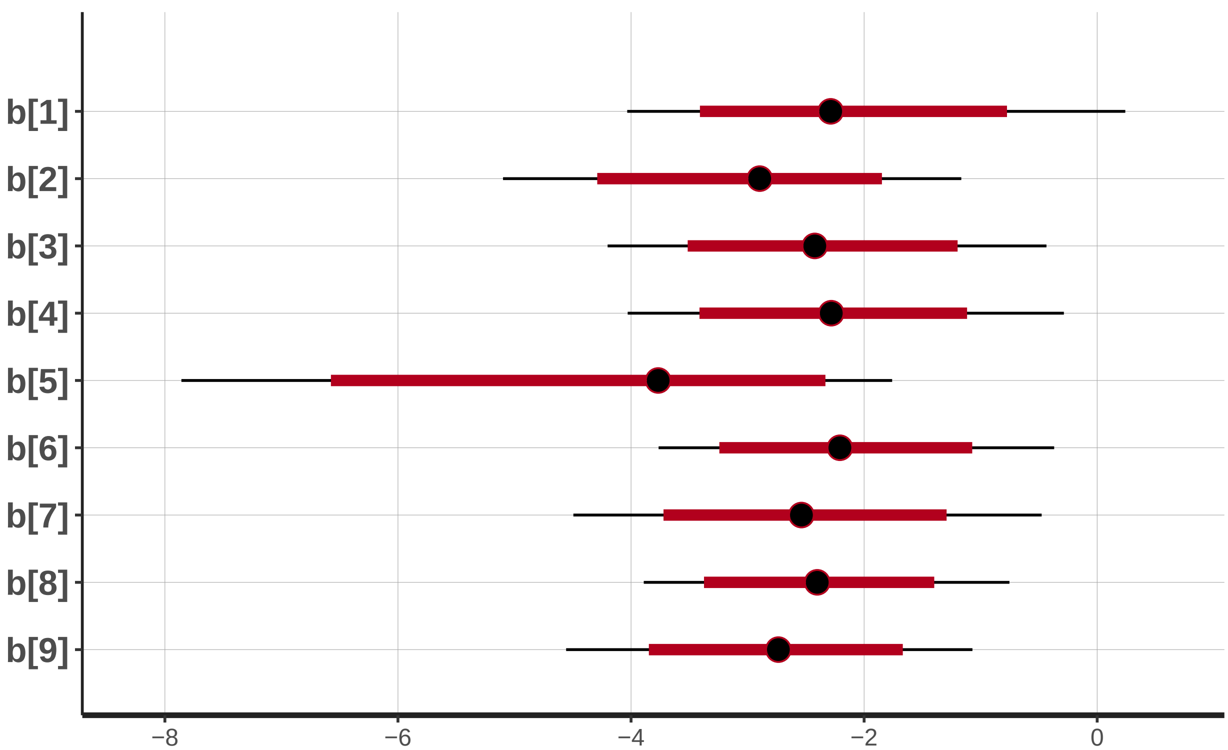
```r
print(fit_partpool, digits=3, probs=c(0.025, 0.975))
```

```
## Inference for Stan model: 09c04efcb9dcbe2facbb0e83ef1aa41b.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##            mean se_mean    sd     2.5%   97.5% n_eff  Rhat
## a[1]      9.346   0.032 1.360    6.621  12.009  1826 1.000
## a[2]     11.861   0.048 1.493    8.528  14.473   956 1.002
## a[3]      4.016   0.032 1.283    1.606   6.608  1653 1.002
## a[4]      4.198   0.041 1.355    1.547   6.855  1089 1.004
## a[5]      9.307   0.048 1.598    6.281  12.505  1106 1.003
## a[6]      4.907   0.032 1.316    2.344   7.536  1736 1.001
## a[7]      4.762   0.031 1.434    1.933   7.574  2161 1.001
## a[8]      5.477   0.041 1.281    2.964   8.130   997 1.004
## a[9]      6.218   0.024 1.305    3.662   8.969  3046 1.001
## b[1]     -2.179   0.028 1.069   -4.032   0.241  1484 1.005
## b[2]     -2.991   0.024 0.978   -5.098  -1.166  1704 1.002
## b[3]     -2.380   0.020 0.933   -4.201  -0.437  2233 1.002
## b[4]     -2.272   0.025 0.930   -4.028  -0.286  1352 1.005
## b[5]     -4.144   0.063 1.637   -7.857  -1.759   679 1.003
## b[6]     -2.175   0.024 0.851   -3.763  -0.369  1227 1.004
## b[7]     -2.518   0.022 0.990   -4.495  -0.471  1971 1.003
## b[8]     -2.375   0.021 0.776   -3.891  -0.753  1341 1.004
## b[9]     -2.735   0.020 0.877   -4.556  -1.071  1860 1.003
## sigma     2.981   0.014 0.454    2.241   4.052  1097 1.005
## mu_a      6.594   0.024 1.191    4.199   9.010  2455 1.001
## sigma_a   3.169   0.034 1.103    1.417   5.767  1060 1.003
## mu_b     -2.626   0.017 0.639   -3.880  -1.426  1427 1.006
## sigma_b   1.052   0.033 0.706    0.187   2.733   472 1.004
## lp__    -89.426   0.483 6.067 -100.416 -74.555   158 1.010
##
## Samples were drawn using NUTS(diag_e) at Tue Oct 12 14:12:38 2021.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```r
plot(fit_partpool, pars="a")
```



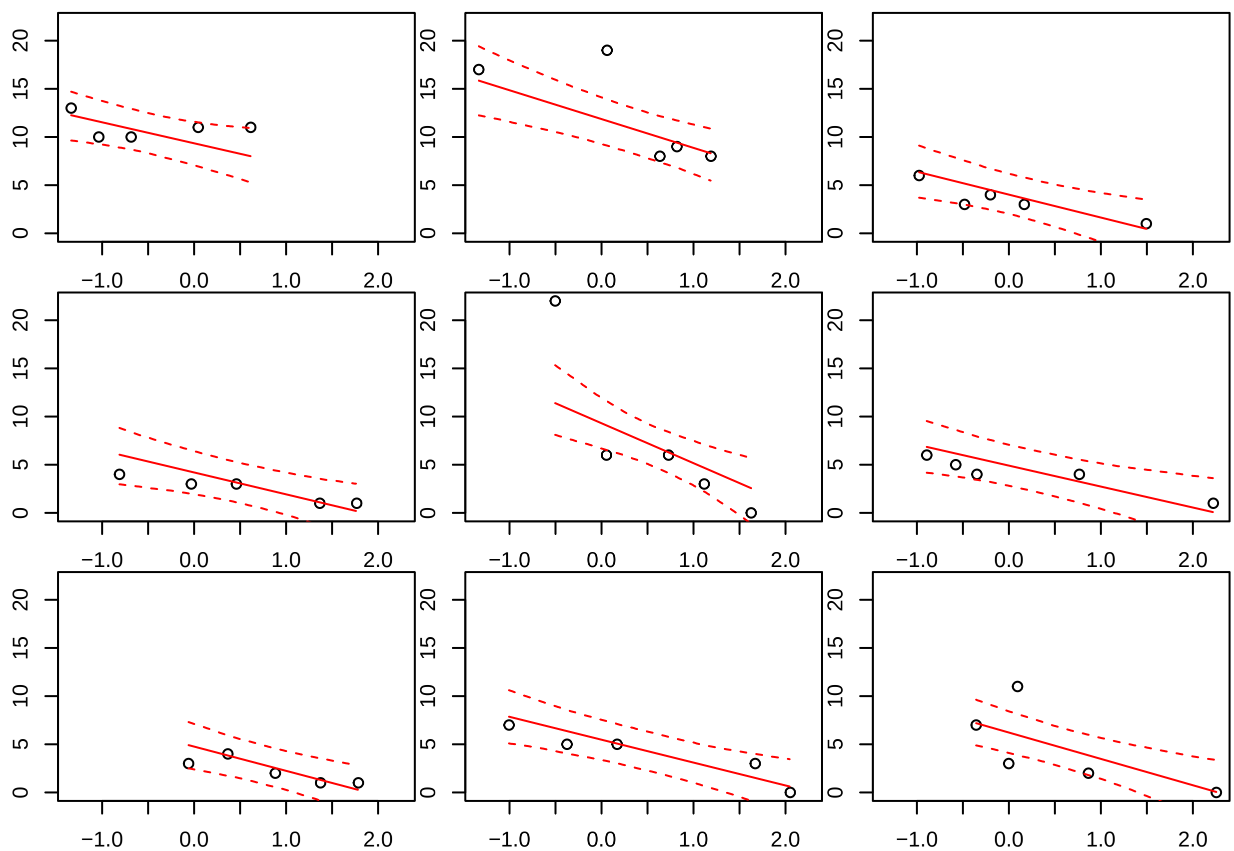

```
plot(fit_partpool, pars="b")
```



Predictions / credible intervals

Again, we can generate predictions and compute credible intervals (for the deterministic part of the model). Here: 90% credible intervals.

```
posterior=as.matrix(fit_partpool)

par(mfrow=c(3,3), mar=c(1,1,1,1), oma=c(1,1,0,0))
for (i in 1:9){
  df.sub=subset(df, df$Beach==i)
  x.pred = seq(from=min(df.sub$NAP), to=max(df.sub$NAP), by=0.01)
  plot(df.sub$NAP,  df.sub$Richness,
       xlim=range(df$NAP),
       ylim=range(df$Richness)  )

  y.cred = matrix(0, nrow=nrow(posterior), ncol=length(x.pred))
  for(j in 1:nrow(posterior)){
    y.cred[j, ] = posterior[j,paste0("a[",i,"]")] + posterior[j,paste0("b[",i,"]")]*x.pred
  }

  y.cred.mean = apply(y.cred, 2, function(x) mean(x))
  lines(x.pred, y.cred.mean, col="red")

  y.cred.q05 = apply(y.cred, 2, function(x) quantile(x, probs=0.05))
  lines(x.pred, y.cred.q05, col="red", lty=2)

  y.cred.q95 = apply(y.cred, 2, function(x) quantile(x, probs=0.95))
  lines(x.pred, y.cred.q95, col="red", lty=2)
}
```



brms complete pooling

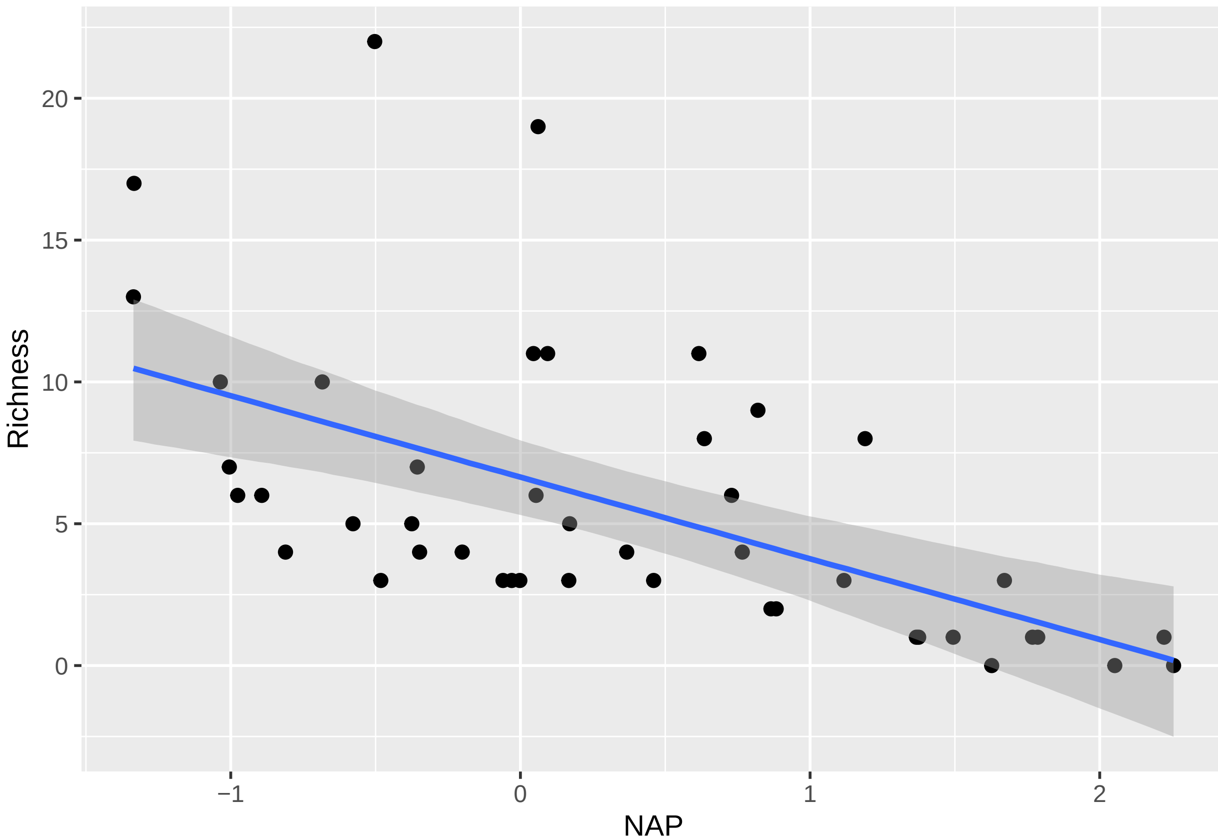
We start with the "complete pooling" model in `brms`, which fits just a linear regression on `NAP`, ignoring the effect of predictor `Beach` on the intercept and slope. It's the same complete pooling model as in the random intercepts regression before.

```
fit.b.compl = brm( Richness ~ NAP,
                   data=df )
```

```
fit.b.compl
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: Richness ~ NAP
##    Data: df (Number of observations: 45)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     6.64      0.67     5.31     7.94 1.00     3382     2541
## NAP          -2.87      0.64    -4.13    -1.60 1.00     3834     2667
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     4.24      0.46     3.46     5.25 1.00     3193     2526
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
plot( conditional_effects(fit.b.compl),
      points=TRUE,
      ask=FALSE )
```



brms no pooling

Next, "no pooling" fits different intercepts and slopes for all level of the categorical predictor `Beach`.

But first we must code `Beach` as a factor, otherwise it would be interpreted as a continuous predictor.

```
str(df)
```

```
## 'data.frame':    45 obs. of  5 variables:
##  $ Sample  : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Richness: int  11 10 13 11 10 8 9 8 19 17 ...
##  $ Exposure: int  10 10 10 10 10 8 8 8 8 8 ...
##  $ NAP     : num  0.045 -1.036 -1.336 0.616 -0.684 ...
##  $ Beach   : int  1 1 1 1 1 2 2 2 2 2 ...
```

```
df$Beach = as.factor(df$Beach)
```

```
fit.b.no = brm( Richness ~ NAP * Beach,
                data=df )
```

As usual with categorical predictors, `lm()` or `brm()` uses dummy coding. I.e. the presented effects are not the different intercepts and slopes for all levels of `Beach`. The intercept for the first level of `Beach` is `Intercept`, but `Beach2` etc are the differences in intercept for the other levels. `NAP` (effect name `b_NAP`) is the slope for the first level of `Beach`, and `NAP:Beach2` etc are the differences in slope for the other levels.

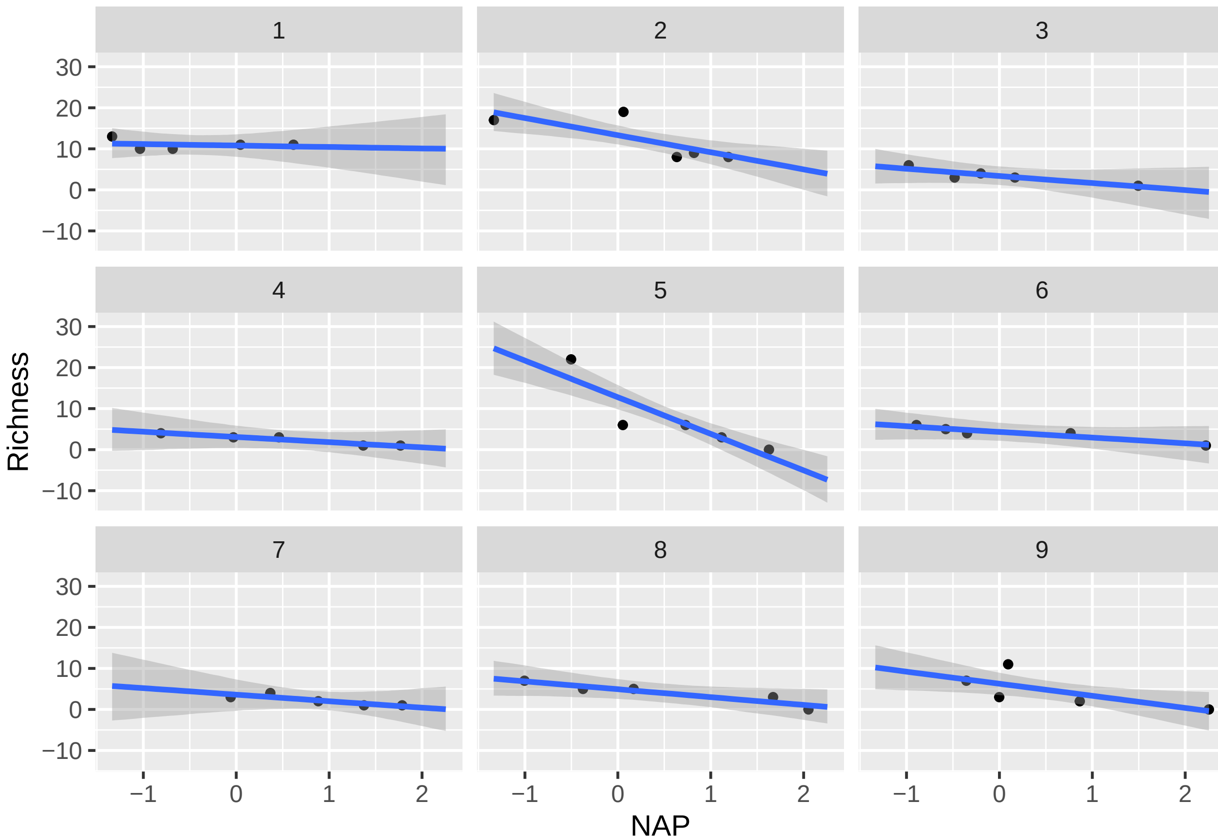
```
fit.b.no
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: Richness ~ NAP * Beach
```

```
##    Data: df (Number of observations: 45)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Population-Level Effects:
##            Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     10.78      1.42     8.04    13.53 1.00      992     1576
## NAP           -0.39      1.60    -3.53     2.64 1.00      892     1480
## Beach2         2.56      1.85    -1.00     6.12 1.00     1422     2096
## Beach3        -7.38      1.81   -10.90    -3.82 1.00     1385     1888
## Beach4        -7.69      1.96   -11.55    -3.74 1.00     1498     2209
## Beach5         2.00      2.05    -2.07     6.02 1.00     1752     2229
## Beach6        -6.45      1.82   -10.08    -2.92 1.00     1363     2109
## Beach7        -7.22      2.44   -12.03    -2.36 1.00     1821     2228
## Beach8        -5.84      1.91    -9.60    -2.08 1.00     1376     1955
## Beach9        -4.52      1.96    -8.39    -0.55 1.00     1509     2229
## NAP:Beach2    -3.77      2.03    -7.73     0.24 1.00     1290     2174
## NAP:Beach3    -1.37      2.07    -5.34     2.60 1.00     1262     1998
## NAP:Beach4    -0.87      2.00    -4.74     3.11 1.00     1200     2164
## NAP:Beach5    -8.54      2.23   -12.95    -4.14 1.00     1295     2330
## NAP:Beach6    -1.00      1.87    -4.57     2.67 1.00     1118     1930
## NAP:Beach7    -1.17      2.38    -5.78     3.73 1.00     1462     1829
## NAP:Beach8    -1.52      1.87    -5.09     2.16 1.00     1201     1788
## NAP:Beach9    -2.57      2.03    -6.50     1.55 1.00     1321     2150
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     2.57      0.37     1.96     3.42 1.00     1726     1993
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

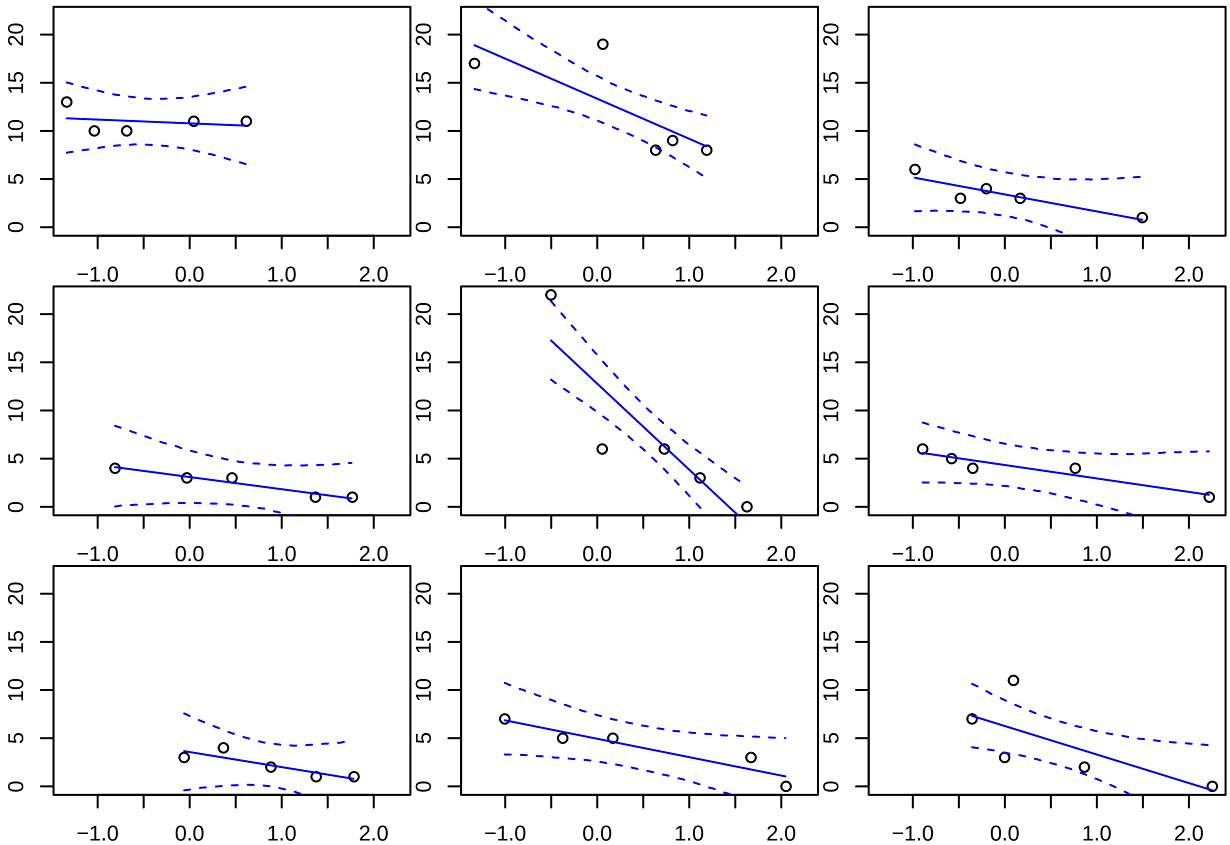
By using `conditions=` and specifying all levels of `Beach`, effect of `NAP` is shown for all levels. Slope is different in all levels.

```r
plot( conditional_effects(fit.b.no,
                          effects = "NAP",
                          conditions = data.frame( Beach=levels(df$Beach) ) ),
      points=TRUE,
      ask=FALSE )
```



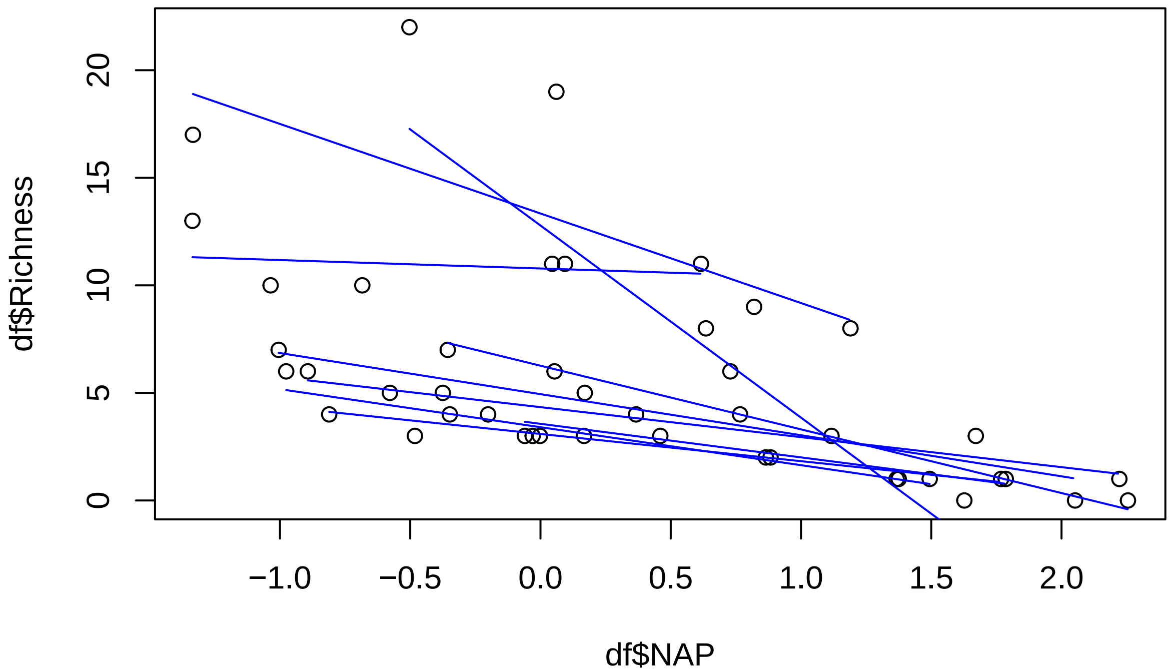
The group-level predictions can also be extracted by hand using `fitted()`. We can specify the range of `NAP` and levels of `Beach` with `newdata`. We plot 95% credible intervals.

```
levels = levels(df$Beach)
par(mfrow=c(3,3), mar=c(1,1,1,1), oma=c(1,1,0,0))

for (i in 1:9){
  df.sub=subset(df, df$Beach==i)
  x.pred = seq(from=min(df.sub$NAP), to=max(df.sub$NAP), by=0.01)
  plot(df.sub$NAP,  df.sub$Richness,
       xlim=range(df$NAP),
       ylim=range(df$Richness)   )

  y.cred = fitted(fit.b.no, newdata=data.frame(NAP=x.pred,
                                               Beach=levels[i] ) )

  lines(x.pred, y.cred[, 1], col="blue")
  lines(x.pred, y.cred[, 3], col="blue", lty=2)
  lines(x.pred, y.cred[, 4], col="blue", lty=2)
}
```



Or we can plot the whole dataset with all group-level mean regression lines.

```
plot(df$NAP, df$Richness)

for (i in 1:9){
  df.sub=subset(df, df$Beach==i)
  x.pred = seq(from=min(df.sub$NAP), to=max(df.sub$NAP), by=0.01)
  y.cred = fitted(fit.b.no, newdata=data.frame(NAP=x.pred,
                                               Beach=levels[i] ) )
  lines(x.pred, y.cred[, 1], col="blue")
}
```



brms partial pooling

With partial pooling, we use a random intercepts and slopes for the categorical predictor `Beach`. This time, we provide priors for the "fixed effects", i.e. mean intercept and slope.

```
priors = c(prior(normal(5,5), class=Intercept),
           prior(normal(0,10), class=b) )
```

```
fit.b.part = brm( Richness ~ 1+NAP + (1+NAP|Beach),
                  prior=priors,
                  data=df )
```

Now, in the population-level effects mean intercept μ_a and slope μ_b are shown. The group-level effects only contain a summary of the random effects. It's standard deviations (σ_a, σ_b) and the correlation between slopes and intercepts cor(a, b).

```
fit.b.part
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: Richness ~ 1 + NAP + (1 + NAP | Beach)
##    Data: df (Number of observations: 45)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Group-Level Effects:
## ~Beach (Number of levels: 9)
##                   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)         3.83      1.20     2.05     6.73 1.00      994     1671
## sd(NAP)               1.86      0.89     0.29     3.92 1.00     1006     1174
## cor(Intercept,NAP)   -0.62      0.34    -0.99     0.28 1.00     1951     2401
##
## Population-Level Effects:
```

```
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     6.63      1.36     3.91     9.30 1.00     1208     1671
## NAP          -2.71      0.83    -4.39    -1.07 1.00     1682     2019
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     2.73      0.41     2.06     3.65 1.00     1226     2086
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Again, instead of fitting all intercepts a_j and slopes b_j `brm()` fits the differences $\alpha_j = a_j - \mu_a$ and $\beta_j = b_j - \mu_b$. The model reads

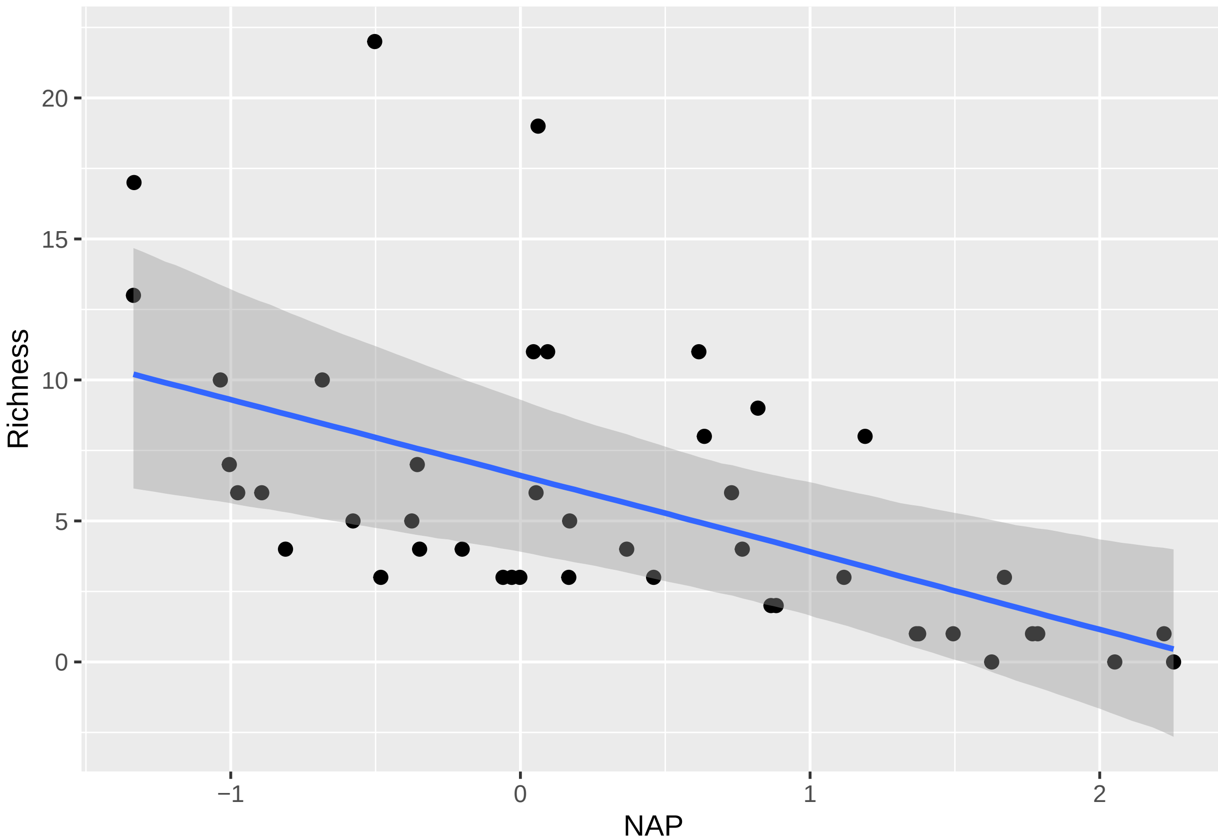
$$
\begin{aligned}
y_i &\sim \text{normal}((\mu_a + \alpha_{group(i)}) + (\mu_b + \beta_{group(i)}) \cdot x, \sigma), \quad i = 1, ..., n \quad (n \text{ observations}) \\
\alpha_j &\sim \text{normal}(0, \sigma_a), \quad j = 1, ..., m \quad (m \text{ groups}) \\
\beta_j &\sim \text{normal}(0, \sigma_b), \quad j = 1, ..., m \quad (m \text{ groups})
\end{aligned}
$$

```
ranef(fit.b.part)
```

```
## $Beach
## , , Intercept
##
##    Estimate Est.Error       Q2.5     Q97.5
## 1  2.540847  1.743882 -0.8419498 6.0209580
## 2  5.788072  1.833344  2.3432109 9.3839597
## 3 -2.804893  1.683489 -6.0873369 0.4444730
## 4 -2.926661  1.800657 -6.4835468 0.5165117
## 5  4.207380  2.045448  0.1887037 8.2839585
## 6 -2.038257  1.654506 -5.3573790 1.2476575
## 7 -2.323106  1.935573 -6.3232722 1.4981540
## 8 -1.432882  1.745339 -4.7384368 2.0101851
## 9 -0.329528  1.759999 -3.7325519 3.1625538
##
## , , NAP
##
##      Estimate Est.Error      Q2.5      Q97.5
## 1  0.14891816  1.255007 -2.074871 3.01373291
## 2 -1.51070274  1.305144 -4.210327 0.86520672
## 3  0.88227617  1.140999 -1.300154 3.30303108
## 4  1.01779069  1.179102 -1.200473 3.52142295
## 5 -3.02565058  1.833075 -6.692674 0.02281932
## 6  0.92695921  1.071852 -1.093888 3.23927093
## 7  0.67046217  1.309826 -1.895710 3.41201102
## 8  0.58224544  1.046994 -1.325798 2.73604016
## 9 -0.07492481  1.098433 -2.286378 2.17586757
```

Now, let's look at observed and predicted. `conditional_effects()` plots fixed effects only by default. The whole dataset is shown and predictions with mean slope μ_b and mean intercept μ_a.

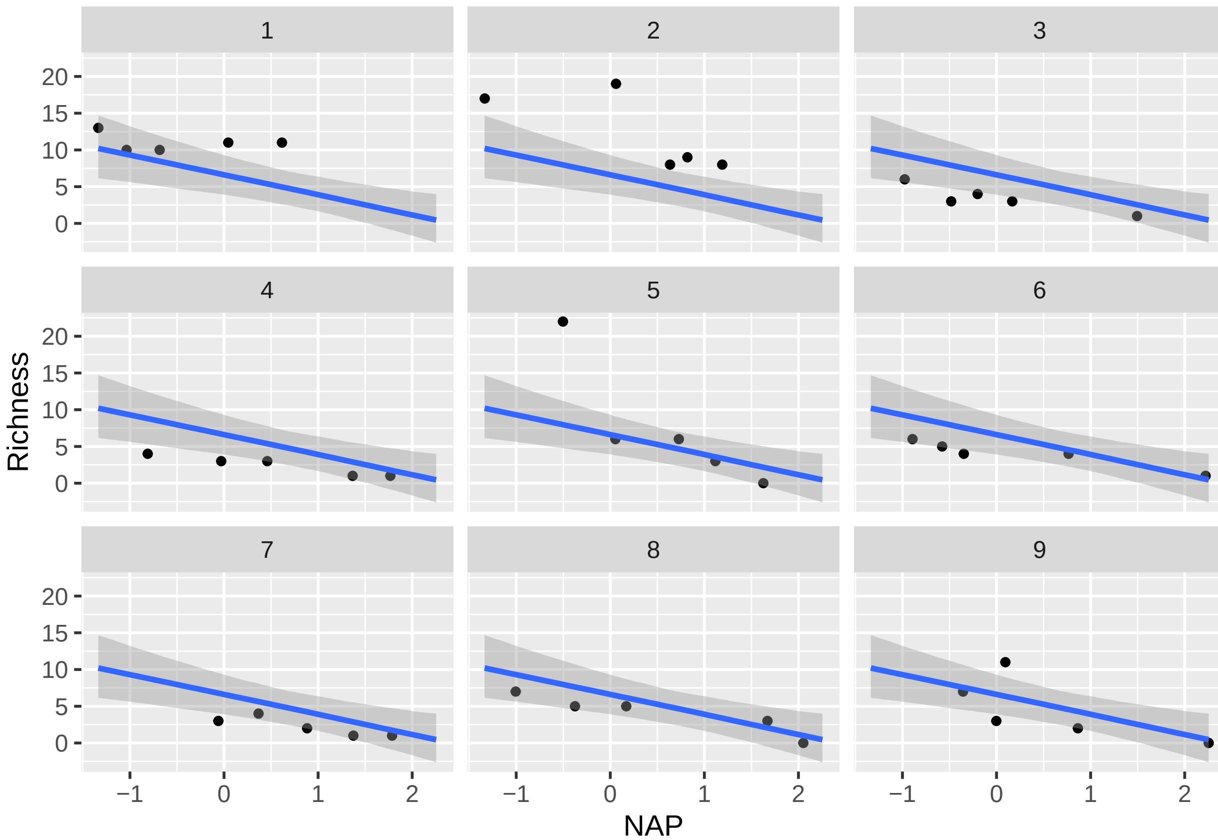
```
plot( conditional_effects(fit.b.part),
      points=TRUE,
      ask=FALSE )
```



When specifying group-level predictors (all levels) with `conditions=`, we receive a warning that they are not part of the (fixed effects) model. We see aplot for all levels of `Beach`, but the regression line is the same: predictions for fixed effects part only!

```
plot( conditional_effects(fit.b.part,
                          effects = "NAP",
                          conditions = data.frame(Beach=levels(df$Beach)) ),
      points=TRUE,
      ask=FALSE )
```

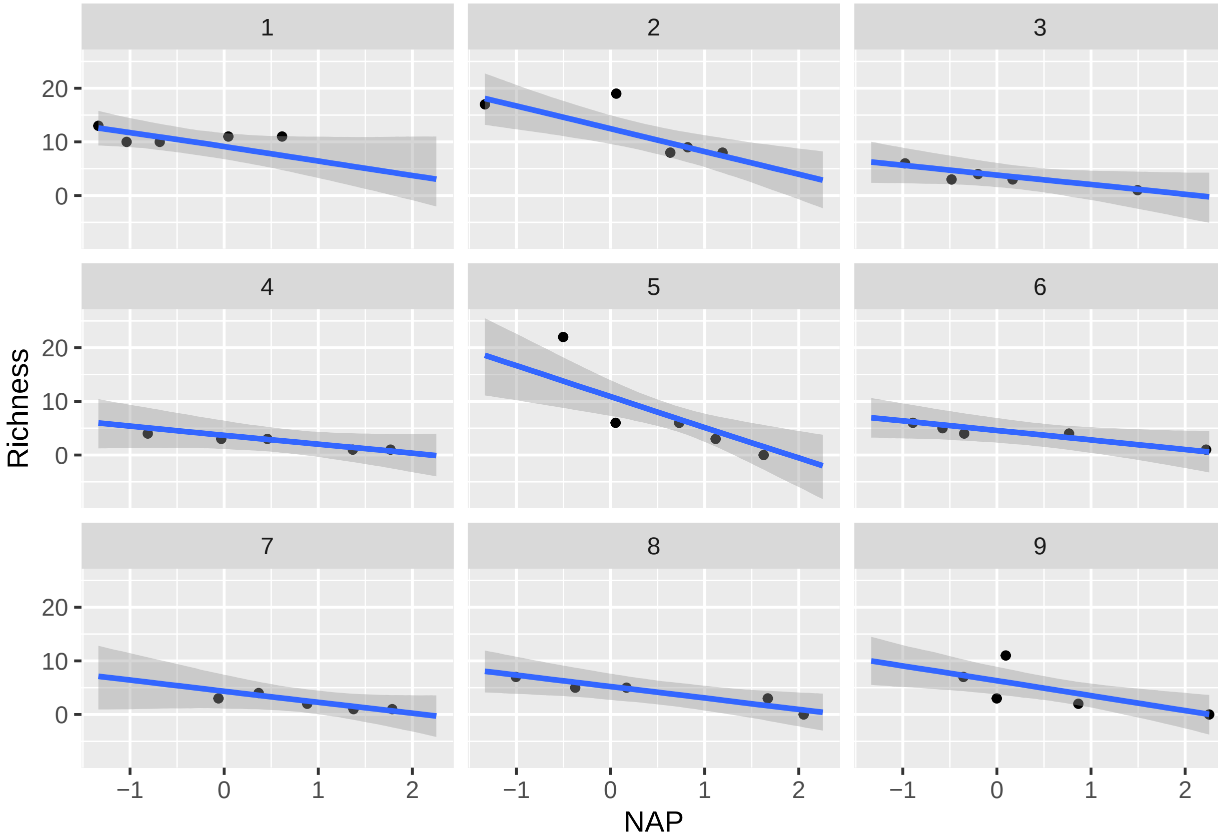
```
## Warning: The following variables in 'conditions' are not part of the model:
## 'Beach'
```



To include also the random effects for model predictions, `re_formula=NULL` must be specified. This reads a little weird (I would have expected something like `re_formula=TRUE`), but the default for no random effects as above is `re_formula=NA` and `NULL` is the command for using random effects for prediction here.

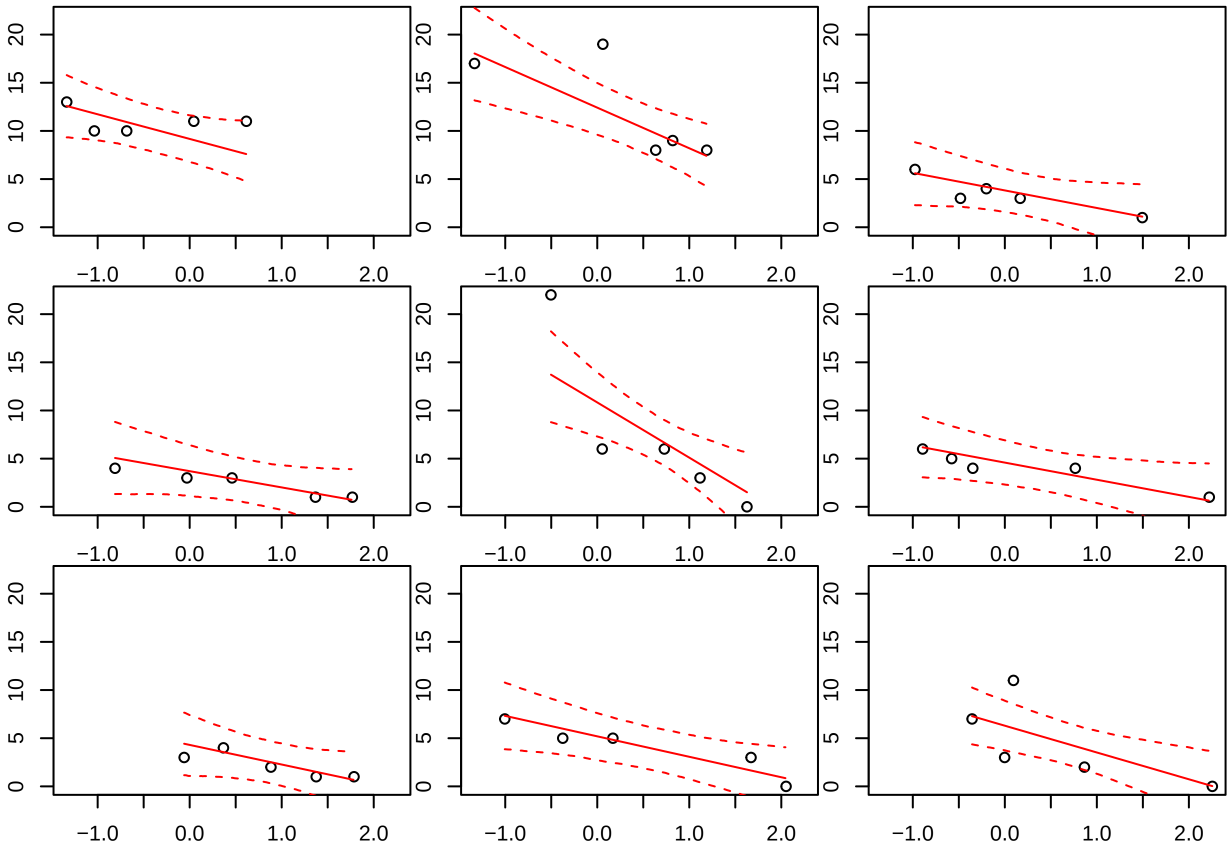
Slopes are identical, but intercepts vary between groups.

```
plot( conditional_effects(fit.b.part,
                          effects = "NAP",
                          re_formula = NULL,
                          conditions = data.frame(Beach=levels(df$Beach)) ),
      points=TRUE,
      ask=FALSE )
```



Same as in the “no pooling” model, group-level predictions can also be extracted by hand using `fitted()`.

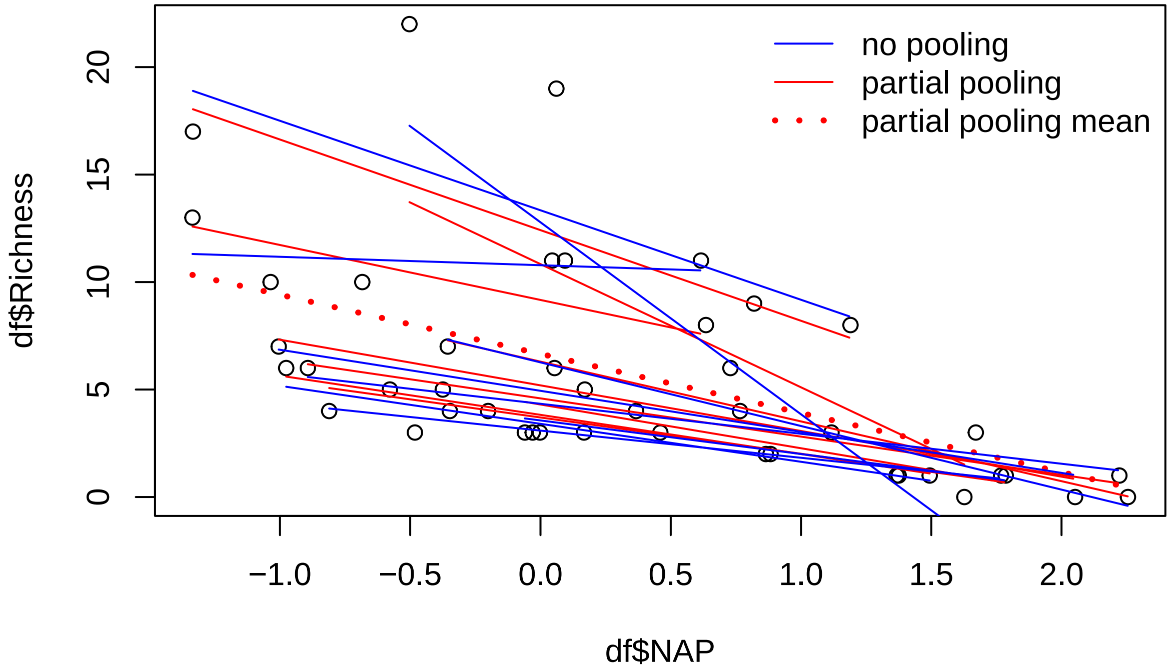
```
levels = levels(df$Beach)

par(mfrow=c(3,3), mar=c(1,1,1,1), oma=c(1,1,0,0))
for (i in 1:9){
  df.sub=subset(df, df$Beach==i)
  x.pred = seq(from=min(df.sub$NAP), to=max(df.sub$NAP), by=0.01)
  plot(df.sub$NAP,  df.sub$Richness,
       xlim=range(df$NAP),
       ylim=range(df$Richness)  )

  y.cred = fitted(fit.b.part, newdata=data.frame(NAP=x.pred,
                                                 Beach=levels[i] ) )

  lines(x.pred, y.cred[, 1], col="red")
  lines(x.pred, y.cred[, 3], col="red", lty=2)
  lines(x.pred, y.cred[, 4], col="red", lty=2)
}
```



We can plot the whole dataset with all group-level mean regression lines, both for the previous "no pooling" model and the current "partial pooling" model. With partial pooling, slopes b_j are drawn towards the overall mean intercept μ_b (i.e. less extreme) compared to no pooling.

```
plot(df$NAP, df$Richness)

# partial pooling
for (i in 1:9){
  df.sub=subset(df, df$Beach==i)
  x.pred = seq(from=min(df.sub$NAP), to=max(df.sub$NAP), by=0.01)
  y.cred = fitted(fit.b.part, newdata=data.frame(NAP=x.pred,
                                                 Beach=levels[i] ) )
  lines(x.pred, y.cred[, 1], col="red")
}

x.pred = seq(from=min(df$NAP), to=max(df$NAP), by=0.01)
y.cred = fitted(fit.b.part, newdata=data.frame(NAP=x.pred,
                                               Beach=NA) )

# partial pooling mean
lines(x.pred, y.cred[, 1], col="red", lwd=3, lty=3)

# no pooling (previous model)
for (i in 1:9){
  df.sub=subset(df, df$Beach==i)
  x.pred = seq(from=min(df.sub$NAP), to=max(df.sub$NAP), by=0.01)
  y.cred = fitted(fit.b.no, newdata=data.frame(NAP=x.pred,
```

```
                                                    Beach=levels[i] ) )
  lines(x.pred, y.cred[, 1], col="blue")
}

legend("topright",
       legend=c("no pooling","partial pooling", "partial pooling mean"),
       lwd=c(1,1,3),
       lty=c(1,1,3),
       col=c("blue","red","red"),
       bty="n")
```



Extra: GLMM

We assumed normally distributed residuals and the default in `brms` is Gaussian indeed. But the reponse we model (species richness) is an integer (not continuous), it's also non-negative. So we could use a Poisson distribution for the stochastic part of the model, which is standard practice for count data. By specifying `family=poisson`, also a log-link is chosen by default. This means the regression lines become an exponential functions, which are non-negative, too.

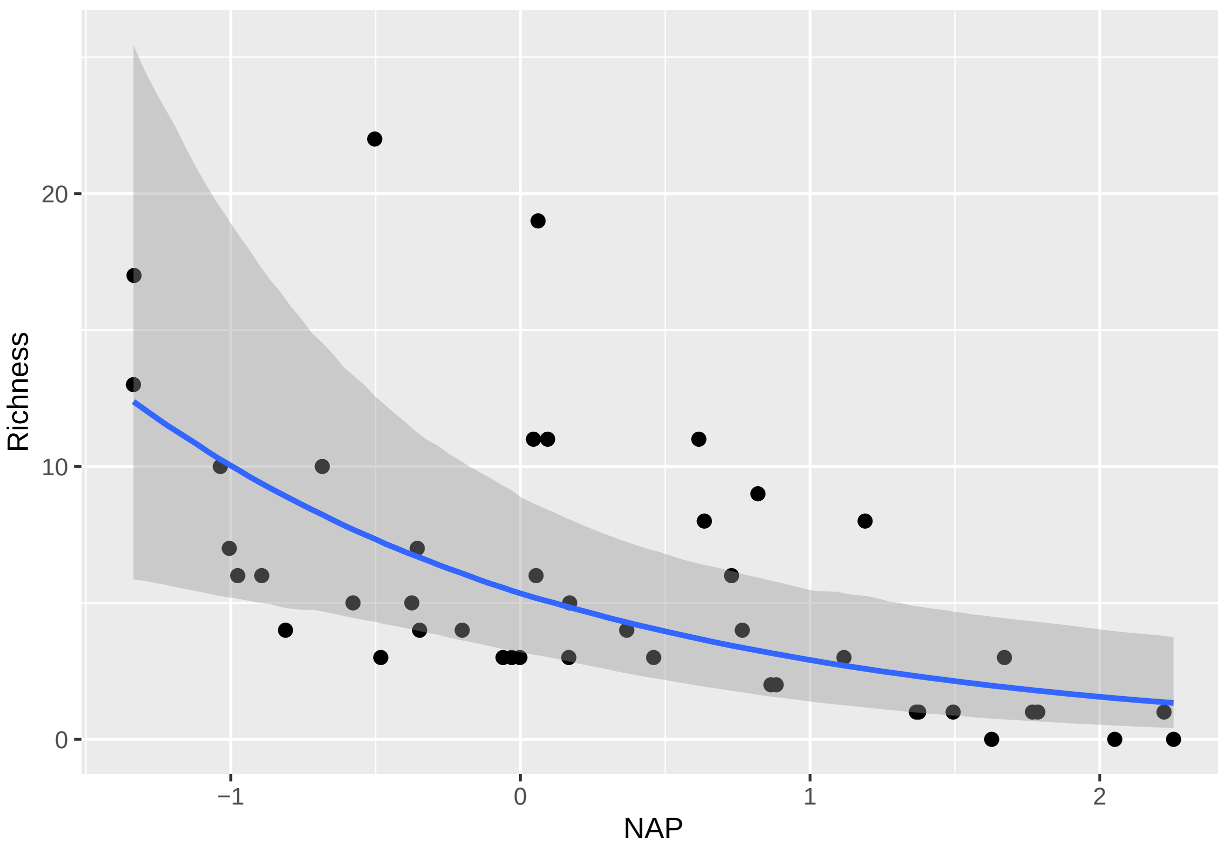
```
fit.b.GLMM = brm( Richness ~ 1+NAP + (1+NAP|Beach),
                  family=poisson,
                  data=df )
```

```
fit.b.GLMM
```

```
## Warning: There were 16 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help. See http://mc-stan.org/misc/
## warnings.html#divergent-transitions-after-warmup

##  Family: poisson
##   Links: mu = log
## Formula: Richness ~ 1 + NAP + (1 + NAP | Beach)
##    Data: df (Number of observations: 45)
```

```
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Group-Level Effects:
## ~Beach (Number of levels: 9)
##                       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)             0.70      0.24     0.38     1.29 1.00     1315     2129
## sd(NAP)                   0.45      0.22     0.14     0.98 1.00      908      906
## cor(Intercept,NAP)        0.11      0.39    -0.67     0.81 1.00     2094     2171
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     1.68      0.26     1.16     2.18 1.00     1203     1597
## NAP          -0.62      0.20    -1.03    -0.24 1.01     1236      851
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
plot( conditional_effects(fit.b.GLMM),
      points=TRUE,
      ask=FALSE )
```




```
p = plot( conditional_effects(fit.b.GLMM,
                              effects = "NAP",
                              re_formula = NULL,
                              conditions = data.frame(Beach=levels(df$Beach)) ),
          points=TRUE,
```

```
          ask=FALSE,
          plot=FALSE)
p[[1]] + ggplot2::ylim(c(0,25))
```