Linear models

Quantification at work: grand mean models

Reconsider Andrew and Jane. They were faced with the problem that potential competitors could challenge the claim "rent a car in 99 seconds" and drag them to court. More precisely, the question was: "will users on average be able . . . ", which is not about individual users, but the *population mean*. A statistical model estimating just that we call a *grand mean model* (GMM). The GMM is the most simple of all models, so in a way, we can think of it as the "grandmother of all models". Although its is the simplest of all, it is of useful application in design research; here are a few more examples:

- with medical infusion pump the frequency of decimal input error (giving the tenfold or the tenth of the prescribed dose) must be below a bearable level
- the checkout process of an e-commerce website must have a a cancel rate not higher than . . .
- the brake lights of a car must be designed to let the following driver react in a certain time frame

The GMM predicts the expected level of performance when the only thing you know is the population someone is from. Prediction here means that the estimated grand mean we take as a best guess for the population average, that is all realizations we have *not* observed. So, it is a generalization regarding all people we have not invited to the lab, but also potential performance differences by situation, for example the daily shape people are in or their current level of motivation. Just consider the many ways people differ in abilities, experience, strategies, preferences, situations, wishes etc. All these differences may also vary for the same person from day to day and influence performance. All these aspects have not been recorded in the Sec99 study and are therefore not taken into account for prediction. All linear models capture the unpredicted variation in a separate parameter σ, the *residual standard deviation*. σ measures the amount of variance that is left after subtracting all explanatory variables. Formally, the likelihood and random formulas of the grand mean model are written with the following likelihood and random term:

$$\mu = \beta_0 y_i \sim \text{Norm}(\mu_i, \sigma_\epsilon)$$

The larger σ is, the more questionnable it is that the grand mean is truly *representative* for the population. From the GM we will depart in to directions. First, in the remainder of this chapter, we will add further predictors to the model, for example age of participants or a design condition. These models will still make statements on population averages, although in a more detailed way. In the following chapter on mixed-effects models, individuals will get spot light. Still, what all the advanced models have in common is that they move variance in the outcome variable from the error variance to predictors. An optimist would say: today's error is tomorrow's predictor.

So, when estimating the grand mean model, we estimate the intercept β_0 and the standard deviation of the Gaussian distributed error term $\sigma\epsilon$. In R, the analysis of the 99 seconds problem unfolds as follows: completion times (ToT) are stored in a data frame, with one observation per row. This data frame is send to the R command `stan_glm` for estimation, using `data = Ver20`. As the `stan_glm` command applies to a huge variety of regression model, the desired model needs further specification. For that purpose, R has its own formula language. The formula of the grand mean model is `ToT ~ 1`. Left of the `~` (*tilde*) operator is the outcome variable. The right hand side specifies the deterministic part. The 1 here has nothing to do with the natural number neighboured by 0 and 2. In R's formula language it represents the *intercept*.

```r
attach(Sec99)
```

```r
M_1 <- stan_glm(ToT ~ 1, data = Ver20)
```

```r
summary(M_1)
```

```
##
## Model Info:
##
##  function:     stan_glm
##  family:       gaussian [identity]
##  formula:      ToT ~ 1
##  algorithm:    sampling
##  priors:       see help('prior_summary')
##  sample:       4000 (posterior sample size)
##  observations: 100
##  predictors:   1
##
## Estimates:
##                 mean   sd     2.5%   25%    50%    75%    97.5%
## (Intercept)    106.0    3.1   99.9  103.8  106.0  108.1  112.0
## sigma           31.5    2.3   27.5   29.9   31.3   33.0   36.5
## mean_PPD       106.0    4.5   97.2  103.0  106.0  109.0  114.9
## log-posterior -494.3    1.0 -497.1 -494.7 -494.0 -493.6 -493.3
##
## Diagnostics:
##               mcse Rhat n_eff
## (Intercept)   0.1  1.0  2711
## sigma         0.0  1.0  2709
## mean_PPD      0.1  1.0  3258
## log-posterior 0.0  1.0  1382
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

The object `M_1` is the model object created by `stan_glm`. When you call the summary you complex listings that represent different aspects of the regression. These aspects, and more are saved inside the object in a hierarchy of lists. The central result of the estimation is the *posterior distribution (PD)*. It is an array of variables over MCMC runs. We extract the posterior distribution from the model object using the `posterior` command (package: `bayr`).

```r
P_1 <- posterior(M_1)
P_1
```

** tbl_post: 4000 samples in 1 chains

Table 1: Coefficients

model	parameter	type	fixef	entities
M_1	1	fixef	Intercept	1

Table 2: Dispersion

parameter
sigma_resid

The posterior object identifies itself by telling the number of MCMC samples, and the variables contained in the model. In the case here, there is just the intercept (representing the grand mean) and the standard deviation of errors.

Much of the time a researcher doesn't want to deal with the posterior, directly, but desires a brief summary of location and uncertainty. Coefficient tables are frequently used, just like one one shown below.Coefficient tables report the central tendency of every coefficient, which is an indicator for the magnitude of an effect. Next to that, the spread of the posterior distribution is summarized as 95% credibility intervals and represent the degree of uncertainty: the less certain an estimate is, the wider is the interval. A 95% credibility interval gives a range of possible values where you can be 95% certain that it contains the true value. A coefficient table is produced by the `coef` command of the bayr library:

```
coef(P_1)
```

Table 3: Estimates with 95% credibility limits

parameter	type	fixef	center	lower	upper
Intercept	fixef	Intercept	106.0	99.9	112.0
sigma_resid	disp	NA	31.3	27.5	36.5

```
detach(Sec99)
```

Reading coefficient tables

Coefficient tables are the standard way to report regression models. They contain all parameters (or a selection of interest) in rows. For every parameter, the central tendency (center, magnitude, location) is given, and a statement of uncertainty, by convention 95% credibility intervals (CI).

The authors of Bayesian books and the various Bayesian libraries have different opinions on what to report in a coefficient table. Most seem to prefer the posterior mode or the median, only some use the mean.

A disadvantage of the *mean* is that it may change, under many monotonic transformations. A monotonic transformations is a recoding of a variable x_1 into a new variable x_2 by a transformation function ϕ (*phi*) such that the order of values stays untouched. Examples of monotonic functions are the logarithm ($x_2 = \log(x_1)$), the exponential function ($x_2 = \exp(x_1)$), or simply $x_2 = x_1 + 1$. A counter example is the quadratic function $x_2 = x_1^2$. In data analysis monotonous transformations are used a lot. Especially, Generalized Linear Models make use of monotonous link functions to establish linearity @ref(re-linking-linearity).

The *mode* of a distribution is its point of highest density. It is invariant under monotonic transformations. It also has a rather intuitive meaning as the most likely value for the true parameter. Next to that, the mode is compatible with classic maximum likelihood estimation. When a Bayesian takes a pass on any prior information, the posterior mode should precisely match the results of a classic regression engine (e.g., `glm`). The main disadvantage of the mode is that it has to be estimated by one of several heuristic algorithms. These add some computing time and may fail when the posterior distribution is bi-modal. However, when that happens, you probably have a more deeply rooted problem, than just deciding on a suitable summary statistic.

The *median* of a distribution marks the point where half the values are below and the other half are equal or above. Technically, the median is just the 50% quartile of the distribution. The median is extremely easy and reliable to compute, and it shares the invariance of monotonous transformations. This is easy to conceive: The median is computed by ordering all values in a row and then picking the value that is exactly in the middle. Obviously, this values only changes when the order changes, i.e. a non-monotonous function was applied.

For center estimates I use the posterior median by default, for its simplicity. Researchers who desire downward compatibility with classic regression engines, can easily switch to the mode by using the *estimate* argument. The following uses the `shorth` command form the package `modeest`.

```r
attach(Sec99)
```

```r
coef(P_1, estimate = modeest::shorth)
```

Table 4: Estimates with 95% credibility limits

parameter	type	fixef	center	lower	upper
Intercept	fixef	Intercept	106.0	99.9	112.0
sigma_resid	disp	NA	31.1	27.5	36.5

Expressing the level of certainty of the posterior distribution makes statistics *inferential*. When the posterior is widely spread, you will still bet on values close to the center, but keep your bid low. For the spread of a distribution, the standard deviation may come to mind of some readers. The standard deviation has teh disadvantage that a single value does not represent non-symmetric distributions well. A better way is to express certainty as limits, a lower and an upper. The most simple method resembles that of the median by using quantiles. In this book, *2.5% and 97.5% certainty quantiles* are routinely used to form 95% credibility intervals. Again, another method exists to obtain CIs. Some authors prefer to report the *highest posterior interval*, which is the narrowest interval that contains 95% of the probability mass. While this is intriguing to some extent, HPDs are not invariant to monotonic transformations.

The `coef` command by default gives the median and the 2.5% and 97.5% limits. The three parameters have in common that they are quantiles, which are handled by Rs `quantile` command. To demystify the `coef`, here is how you can make a basic coefficient table yourself:

```r
P_1 %>%
  group_by(parameter) %>%
  summarize(center = quantile(value, 0.5),
         lower  = quantile(value, 0.025),
         upper  = quantile(value, 0.975)) %>%
  kable()
```

parameter	center	lower	upper
Intercept	106.0	99.9	112.0
sigma_resid	31.3	27.5	36.5

Note that we get CIs for the dispersion parameter σ, too. Many classic analyses call σ are nuisance parameter and ignore it, or they blame high variation between observations for not reaching "significant" certainty for the parameter of interest. Furthermore, classic regression engines don't yield and measures of certainty on dispersion parameters. I believe that understanding the amount of variation is often crucial for design research and several of the examples that follow try to make the case. This is why we should be glad that Bayesian engines report uncertainty on all parameters involved.

It is common practice to not just drop coefficient tables, but explain and interpret them in written form. My suggestion of how to *report regression results* is to simply walk through the table row-by-row and for every parameter make two statements: a quantitative statement based on the central tendency, and an uncertainty statement. In the present case that would be:

1. The *intercept* β_0 is in the region of 106 seconds, which is pretty off the target of 99 seconds.
2. The certainty is pretty good. At least we can say that the chance of the true mean being 99 seconds or smaller is pretty marginal, as it is not even contained in the 95% CI.

And for σ:

1. The population mean is rather not representative for the observations, as the standard error is almost one third of it.
2. We can be pretty certain that this is so.

```
detach(Sec99)
```

Likelihood and random term

In formal language, regression models are usually specified by *likelihood functions* and one or more *random terms* (exactly one in linear models). The likelihood represents the common, predictable pattern in the data. Formally, the likelihood establishes a link between *predicted values* mu_i and predictors. It is common to call predictors with the Greek letter β (beta). If there are more than one predictors, these are marked with subscripts, starting at zero. The "best guess" is called the *expected value* and is denoted with μ (mu_i). If you just know that the average ToT is 106 seconds and you are asked to guess the performance of the next user arriving in the lab, the reasonable guess is just that, 106 seconds.

$$\mu_i = \beta_0$$

Of course, we would never expect this person to use 106 second, exactly. All observed and imagined observations are more or less clumped around the expected value. The *random term* specifies our assumptions on the pattern of randomness. It is given as a distributions (note the plural), denoted by the $\sim$ (tilde) operator, which reads as: "is distributed". In the case of linear models, the assumed distribution is always the Normal or *Gaussian distribution.* Gaussian distributions have a characteristic bell curve and depend on two parameters: the mean μ as the central measure and the standard deviation σ giving the spread.

$$y_i \sim N(\mu_i, \sigma_\epsilon)$$

The random term specifies how all unknown sources of variation take effect on the measures, and these are manifold. Randomness can arise due to all kinds of individual differences, situational conditions and, last but not least, measurement errors. The Gaussian distribution often is a good approximation for randomness and linear models are routinely used in research. In several classic statistics books, the following formula is used to describe the GMM (and likewise more complex linear models):

$$y_i = \mu_i + \epsilon_i \mu_i = \beta_0 \epsilon_i \sim \text{Norm}(0, \sigma_\epsilon)$$

First, it is to say, that the two ways of formula are mathematically equivalent. The primary difference to our formula is that the *residuals* ϵ_i, are given separately. The pattern of residuals is then specified as a single Normal distribution. Residual distributions are a highly useful concept in modelling as they can be used to check a given model. When residuals are such a highly useful concept, the classic formula is more intuitive. The reason for separating the model into likelihood and random term is that it works in more cases. When turning to Generalized Linear Models (GLM) in chapter @ref(generalized-linear-models), we will use other patterns of randomness, that are no longer additive, like in $\mu_i + \epsilon_i$. As I consider the use of GLMs an element of professional statistical practice, I use the general formula right from the start.

Do the random walk: Markov Chain Monte Carlo sampling

So far, we have seen how linear models are specified and how,parameters are interpreted from standard coefficient table. While it is convenient to have a standard procedure it turns out very useful to understand how these estimates came to life. In Bayesian estimation, the *posterior distribution (PD)* is the central point of departure for any such statements. The PD assigns a degree of certainty for every possible combination of parameter values. In the current case, you can ask the PD, where and how certain the population mean and the residual standard error are, but you can also ask: How certain are we that the population mean is smaller than 99 seconds and σ is smaller than 10?

In a perfect world, we would know the analytic formula of the posterior and derive statements from it. In most non-trivial cases, though, there is no such formula one can work with. Instead, what the regression engine does is to approximate the PD by a random-walk algorithm called Markov-Chain Monte Carlo sampling (MCMC).

In fact, the `stan_glm` command returns a large object that stores, among others, the full random walk. This random walk represent the posterior distribution almost directly. The following code extracts this the posterior distribution from the regression object prints it. When calling the new object (class: tbl_post) directly, it provides a compact summary of all variables in the model, here this is the intercept and the residual standard error.

```
attach(Sec99)
```

```
P_1 <-  posterior(M_1)
P_1
```

** tbl_post: 4000 samples in 1 chains

Table 6: Coefficients

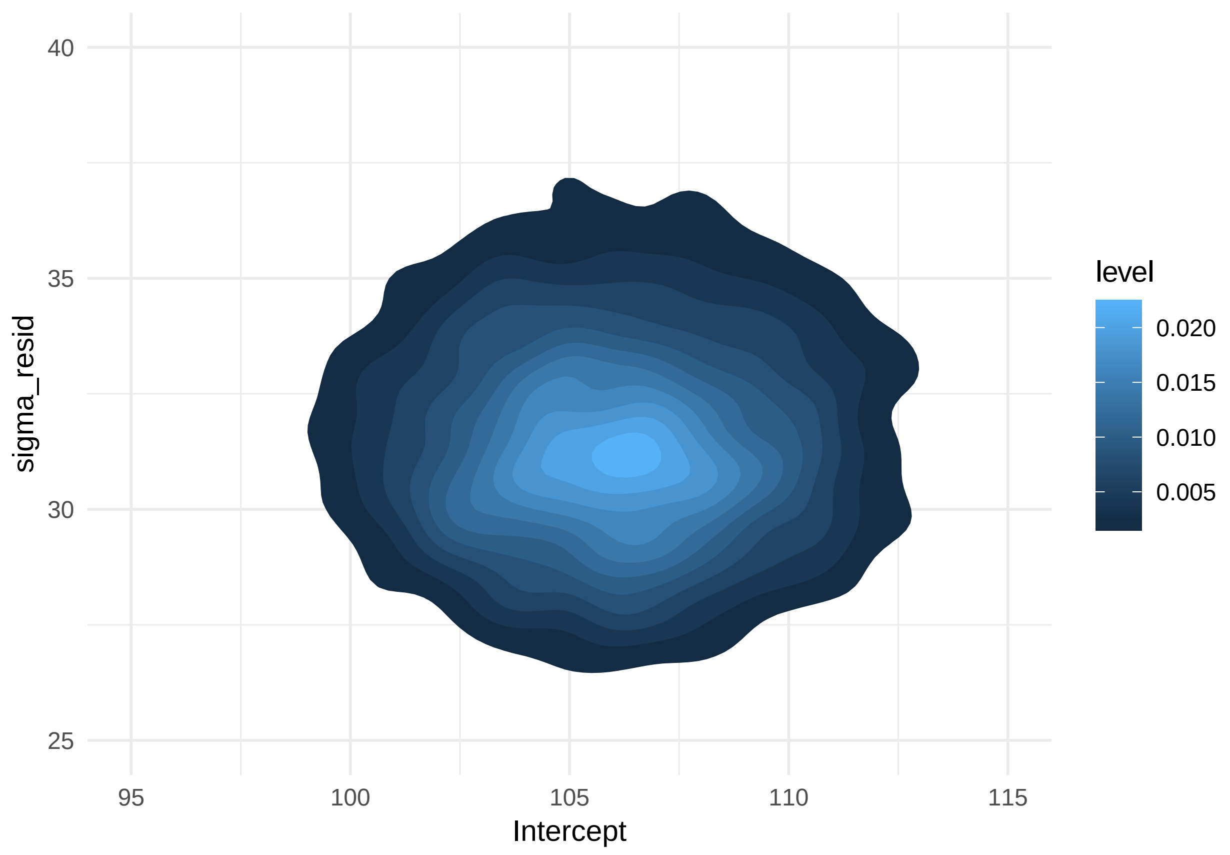
model	parameter	type	fixef	entities
M_1	1	fixef	Intercept	1

Table 7: Dispersion

parameter
sigma_resid

The 99 second GMM has two parameters and therefore the posterior distribution has three dimensions: the parameter dimensions β_0, σ and the probability density. Three dimensional plots are difficult to put on a surface, but for somewhat regular patterns, a density plot with contour lines do a sufficient job:

```
P_1 %>%
  select(chain, iter, parameter, value) %>%
  spread(parameter, value) %>%
  ggplot(aes(x = Intercept, y = sigma_resid, fill = ..level..)) +
  stat_density_2d(geom = "polygon") +
  xlim(95, 115) + ylim(25, 40)
```

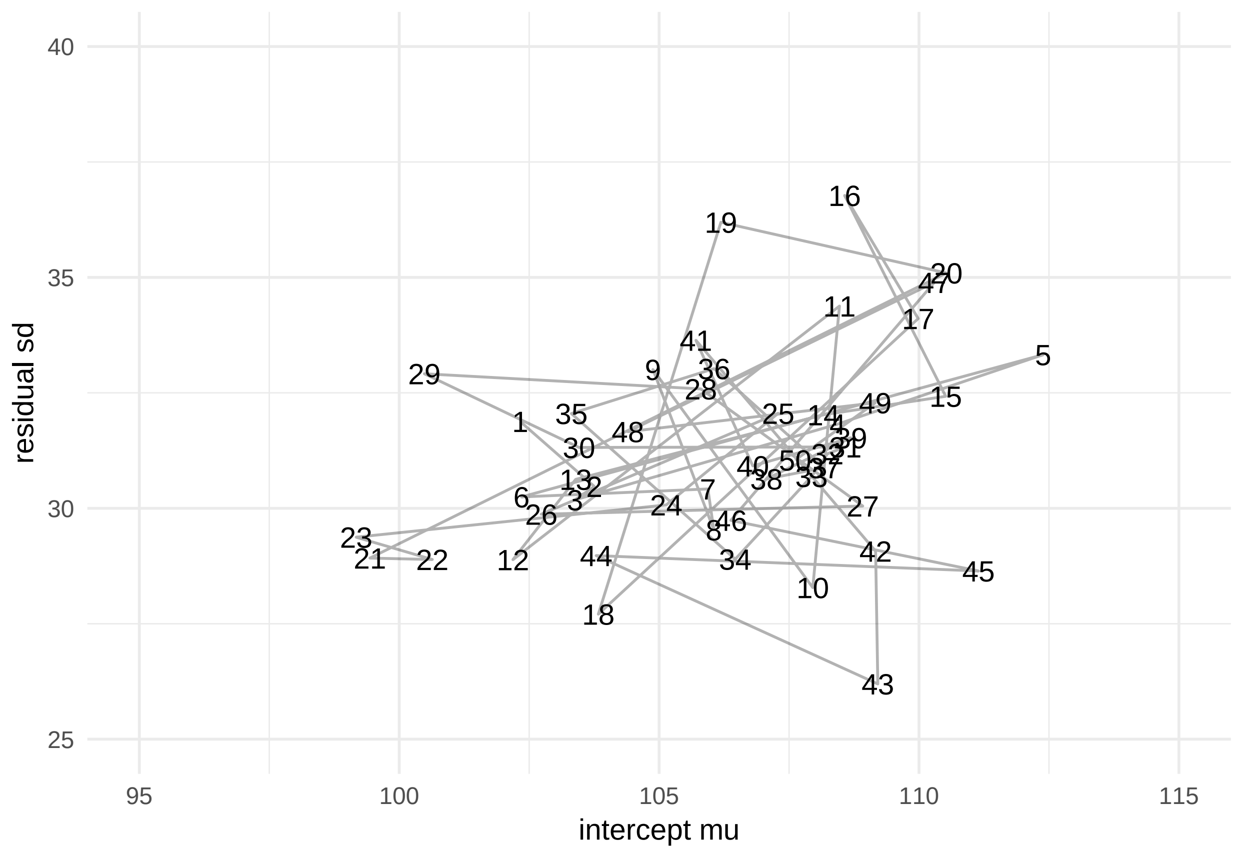


Let's see how this PD "landscape" actually emerged from the random walk. In the current case, the *parameter space* is two-dimensional, as we have μ and σ. The MCMC procedure starts at a deliberate point in parameter space. At every iteration, the MCMC algorithm attempts a probabilistic jump to another location in parameter space and stores the coordinates. This jump is called probabilistic, because it is either carried out, or not, and that depends on a bet. If the new target is in a highly likely region, it is carried out with a higher chance. This sounds circular, but it works and, of course, it has been proven mathematically that it works.

The regression object stores the MCMC results as a long series of positions in parameter space. For any range of interest, it is the relative frequency of visits that represents its certainty. The first 50 hundred steps of the MCMC random walk are shown in @ref(99_seconds_random_walk)'. Apparently, the random walk is not fully random, as the point cloud is more dense in the center area. This is where the more probable parameter values lie. One can clearly see how the MCMC algorithm jumps to more likely areas more frequently. These areas become more dense and, finally, the cloud of visits will approach the contour density plot above.

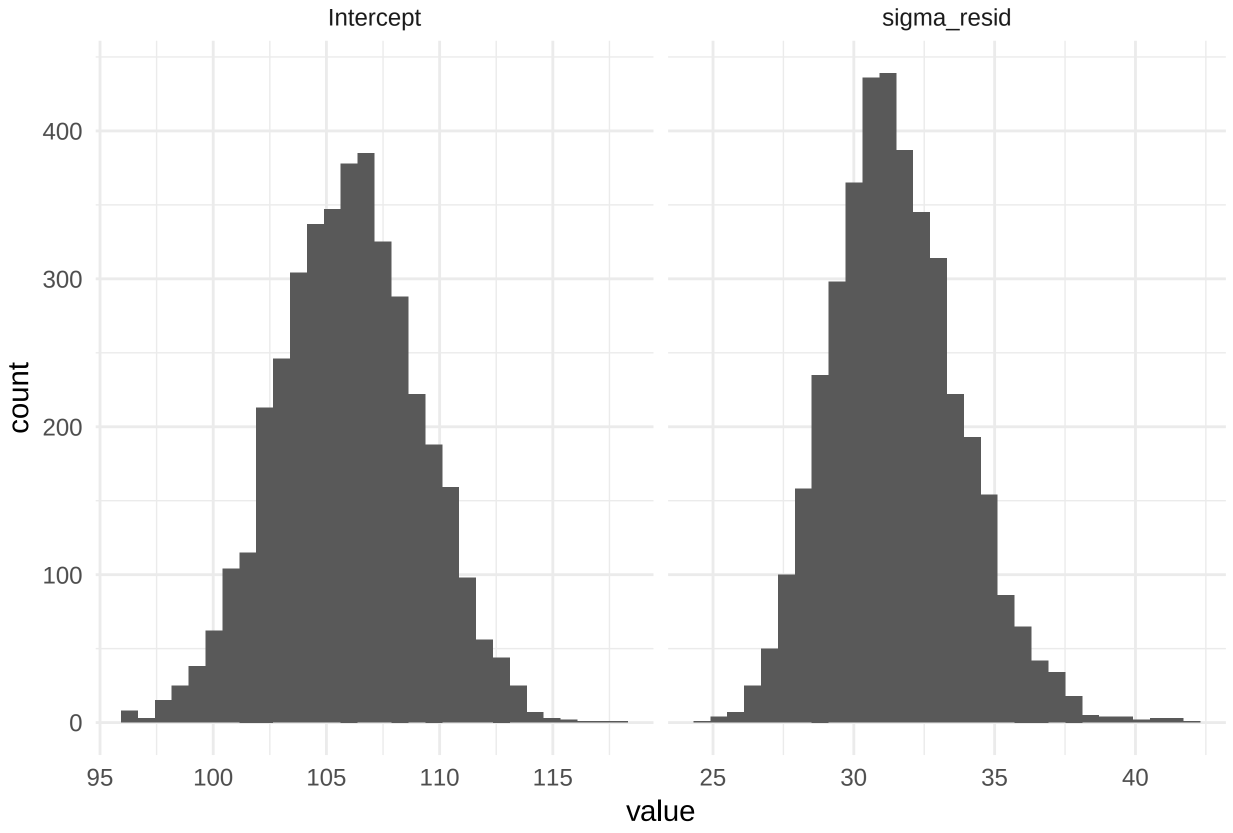
```r
G_random_walk <-
  P_1 %>%
  filter(iter <= 50) %>%
  select(iter, parameter, value) %>%
  spread(parameter, value) %>%
  ggplot(aes(x = Intercept, y = sigma_resid, label = iter)) +
  geom_text() +
  geom_path(alpha = .3) +
  ylab("residual sd") +
  xlab("intercept mu") +
  xlim(95, 115) + ylim(25, 40)
```

```
G_random_walk
```



The more complex regression models grow, the more dimensions the PD gets. The linear regression model in the next chapter has a three parameter dimensions, which is difficult to visualize. Multi-level models have hundreds of parameters, which is impossible to intellectually grasp at once. Therefore, it is common to use the *marginal posterior distributions* (MPD), which give the density of one coefficient at time. My preferred geometry for plotting MPDs is the violin plot, which packs a bunch of densities and therefore can be used for models of higher dimension. Still, for simple models histograms do the job:

```
P_1 %>%
  ggplot(aes(x = value)) +
  geom_histogram() +
  facet_grid(. ~ parameter, scales = "free")
```



In our example, in @ref(99_seconds_post) we can spot that the most likely value for average time-on-task is 106.1. Both distributions have a certain spread. With a wider PD, far-off values have been visited by the MCMC chain more frequently. The probability mass is more evenly distributed and there is less certainty for the parameter to fall in the central region. In the current case, a risk averse decision maker would maybe take the credibility interval as "reasonably certain".

Andrew and Jane expect some scepticism from the marketing people, and some lack in statistical skills, too. What would be the most comprehensible single number to report? As critical decisions are involved, it seems plausible to report the risk to err: how certain are they that the true value is more than 99 seconds. We inspect the histograms. The MPD of the intercept indicates that the average time-on-task is rather unlikely in the range of 99 seconds or better. But what is the precise probability to err for the 99 seconds statement? The above summary with `fixef()` does not answer the question, accurately. The CI gives lower and upper limits for a range of 95% certainty in total. What is needed is the certainty of $\mu \geq 99$. Specific questions deserve precise answers. And once we have understood the MCMC chain as a frequency distribution, the answer is easy: we simply count how many visited values are larger than 99. In R, the `quantile` function handles the job:

```
T_certainty <-
  P_1 %>%
  filter(parameter == "Intercept") %>%
  summarize(certainty_99s = mean(value >= 99),
            certainty_111s = mean(value >= 111))

kable(T_certainty)
```

certainty_99s	certainty_111s
0.986	0.054

It turns out that the certainty for average time-on-task above the 99 is an overwhelming 0.986. The alternative claim, that average completion time is better than 111 seconds, has a rather moderate risk to err (0.054).

```
detach(Sec99)
```

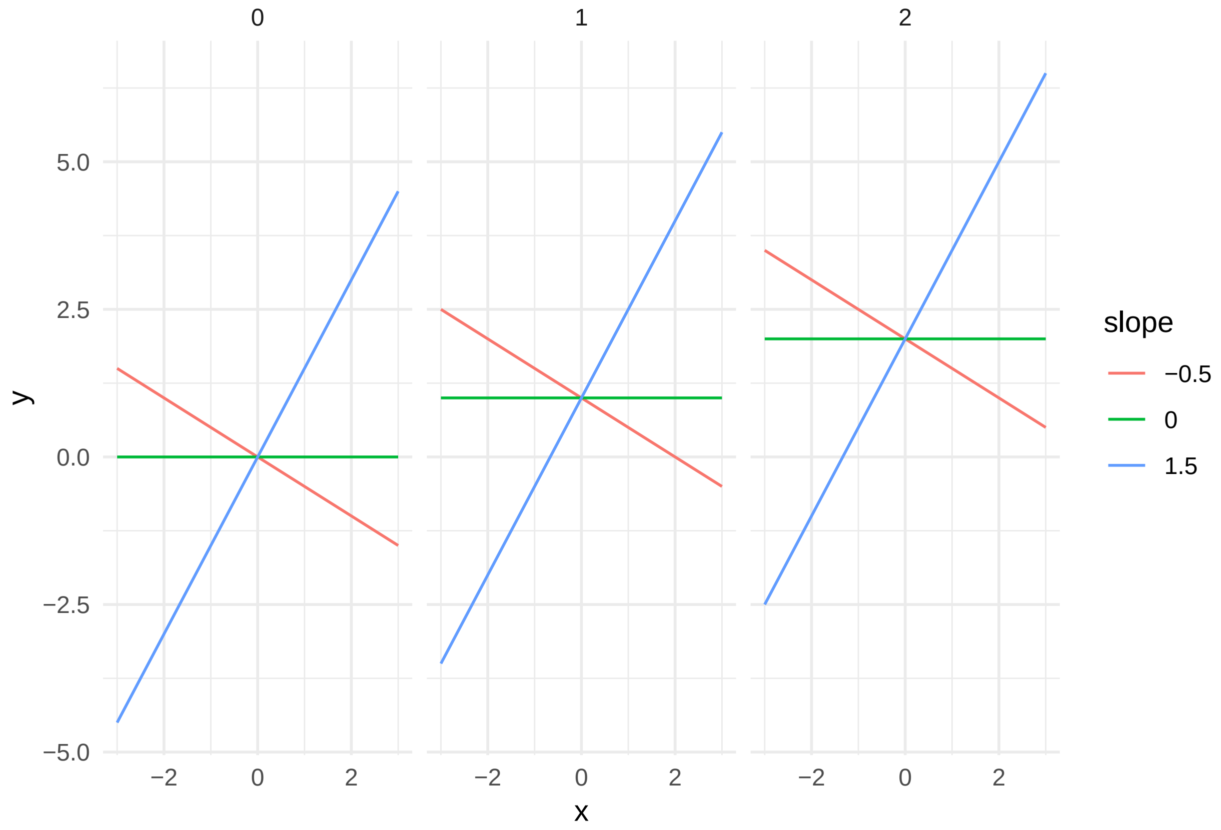
Walk the line: linear regression

In the previous section we have introduced the mother of all regression models: the grand mean model. It assigns rather coarse predictions, without any real predictors. Routinely, design researchers desire to predict performance based on *metric variables*, such as:

- previous experience
- age
- font size
- intelligence level and other innate abilities
- level of self efficiacy, neuroticism or other traits
- number of social media contacts

To carry out such a research question, the variable of interest needs to be measured, next to the outcome variable. And, the variable must vary. You cannot examine the effects of age of font size on reading performance, when all participants are psychology students or you test only one size. Then, for specifying the model, the researcher has to come up with an expectation of how the two are related. Theoretically, that can be any mathematical function, but practically, a *linear function* is often presumed for its simplicity. The following plot shows a variety of linear relations between two variables x and y.

```
mascutils::expand_grid(intercept = c(0, 1, 2),
                       slope = c(-.5, 0, 1.5),
                       x = -3:3) %>%
  arrange(x) %>%
  mutate(y = intercept + x * slope,
         slope = as.factor(slope)) %>%
  ggplot(aes(x = x, y = y, color = slope)) +
  geom_line() +
  facet_grid(~intercept)
```



A linear function is a straight line, which is specified by two parameters: *intercept* β_0 and *slope* β_1:

$$f(x_1) = \beta_0 + \beta_1 x_{1i}$$

The intercept is *"the point where a function graph crosses the x-axis"*, or more formally:

$$f(x_1 = 0) = \beta_0$$

The second parameter, β_1 is called the *slope*. The slope determines the steepness of the line. When the slope is 1, the line will raise by this amount when one moves one step to the right.

$$f(x_1 + 1) = \beta_0 + \beta_1 x_{1i} + \beta_1$$

There is also the possibility that the slope is zero. In such a case, the predictor has no effect and can be left out. Setting $\beta_1 = 0$ produces a horizontal line, with y being constant over the whole range. This shows that The LRM can be conceived a generalization of the GM model: $\mu_i = \beta_0$.

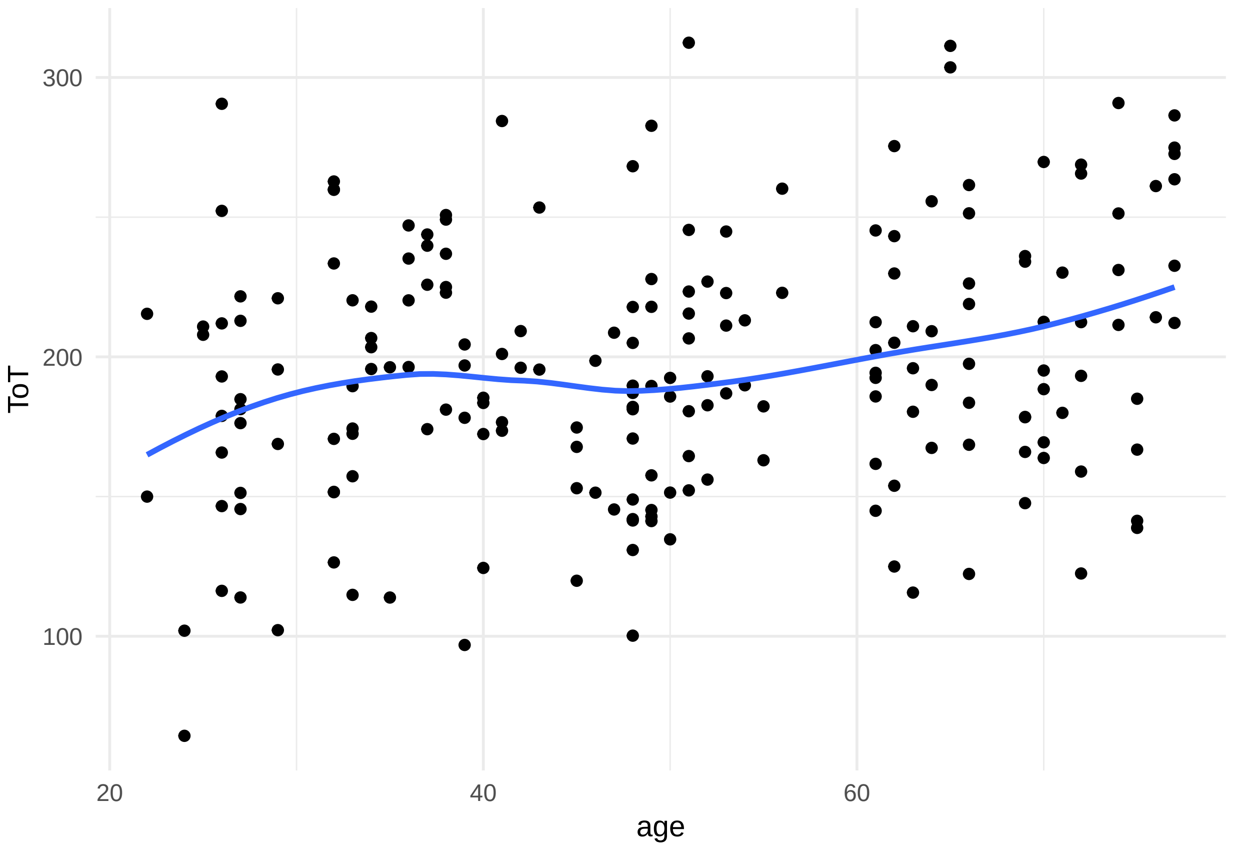
Linear regression gives us the opportunity to discover how ToT can be predicted by age (x_1) in the BrowsingAB case. IN this synthetic experiment, two designs A and B are compared, but this is what we ignore for now. Instead, we ask: are older people slower on the internet? or: is there a linear relationship between age and ToT? The likelihood and random terms of the LRM are:

$$\mu_i = \beta_0 + \beta_1 x_{1i}$$
$$Y_i = N(\mu_i, \sigma)$$

This literally means: with every year of age, ToT increases by the β_1 seconds. Before we run a linear regression with `stan_glm`, we visually explore the association between age and ToT using a scatter plot. The blue line in the graph is a so called a *smoother*, more specifically a LOESS. A smoother is an estimated line, just as linear function. But, it is way more flexible. Where the linear function is a lever fixed at a pivotal point, LOESS is a pipe cleaner. LOESS shows a more detailed picture of the relation between age and ToT. There is a rise between 20 and 40, followed by a stable plateau, and another rise starting at 60. Actually, that does not look like a straight line, but at least there is steady upwards trend.

```
attach(BrowsingAB)
```

```
G_eda_1 <-
  BAB1 %>%
  ggplot(aes(x = age, y = ToT)) +
  geom_point()+
  geom_smooth(se = F, fullrange = F)

G_eda_1
```



In fact, the BrowsingAB data contains what one could call a psychological model. The effect of age is partly due to farsightedness of participants (making them slower at reading), which more or less suddenly kicks in at a certain range of age. Still, we currently make do with a rough linear approximation. To estimate the model, we use the `stan_glm` command in much the same way as before, but add the predictor age. The command will internally check the data type of your variable, which is metric, here. Therefore, it is treated as a metric predictor or *covariate*.

```
M_age <-
  BAB1 %>%
  stan_glm(ToT ~ 1 + age,
           data = .)
```

```
T_age <- fixef(M_age)
T_age
```

Table 9: Estimates with 95% credibility limits

fixef	center	lower	upper
Intercept	164.090	143.152	185.18
age	0.642	0.234	1.04

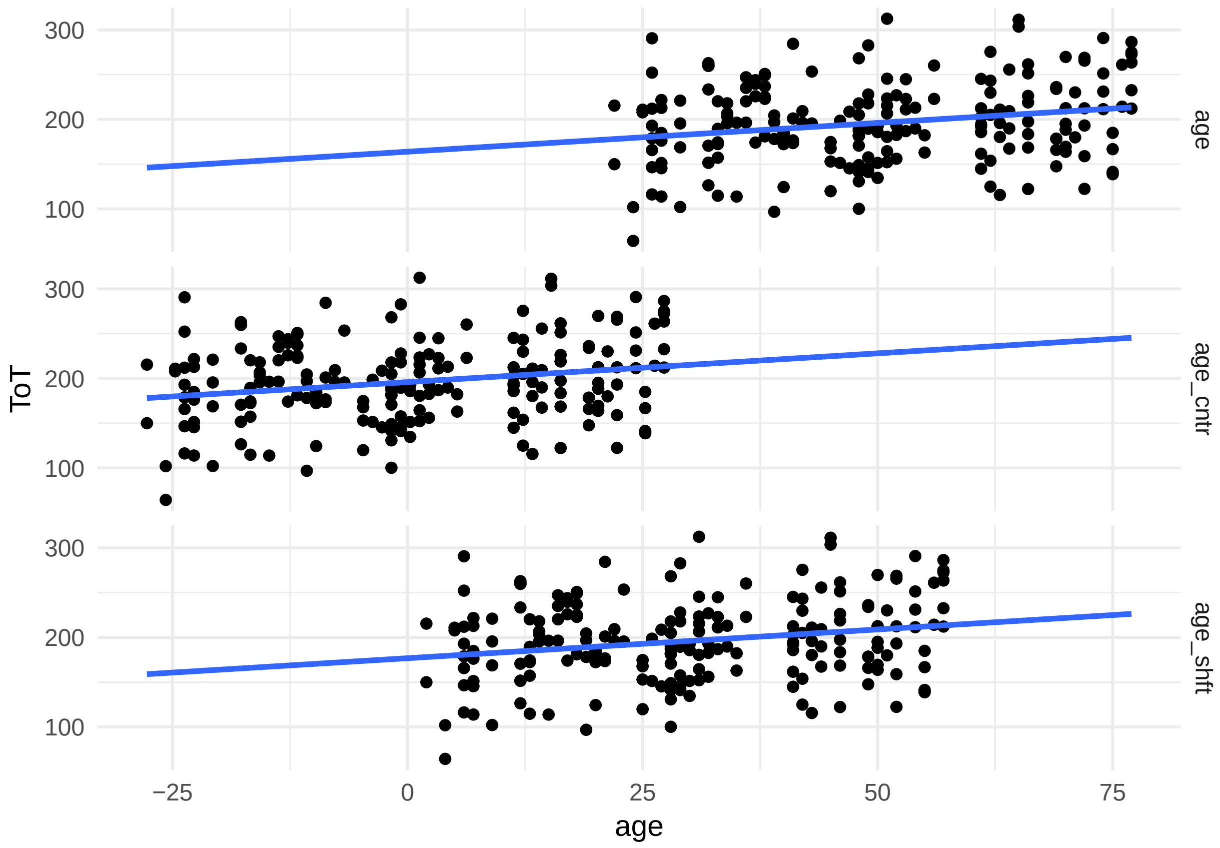
Is age associated with ToT? The coefficient table tells us that with every year of age, users get 0.64 seconds slower, which is considerable. It also tells us that the predicted performance at age = 0 is 164.09.

Transforming variables

The intercept parameter refers to the predicted ToT of a new born. That is a bizarre prediction and we would never seriously put that forward in a stakeholder presentation, or in the conclusion of a scientific paper, would we not? Besides that, the intercept estimate is rather uncertain, with a wide 95% interval, $164.09[143.15, 185.18]_{CI95}$.

Both, implausibility and high certainty are rooted in the same problem: the model puts a parameter, where there is no data. The broad region of the intercept is as empty as the Khali desert, because observations are impossible or have not been recorded. Fortunately, there is a pragmatic solution to the problem: *shifting the predictor*. "Shifting" literally means that the age predictor is moved to the right or the left, such that the zero point is in a region populated with observations. In the case, here, two options seem to make sense: either, the intercept is in the region of youngest participants, or it is the sample average, which is then called *centering*. To shift a variable, just subtract the amount of units (years) where you want the intercept to be. Figure XY shows a shift of -20 and a centering shift on the original variable age

```
BAB1 <-
  BAB1 %>%
  mutate(age_shft = age - 20,
         age_cntr = age - mean(age))

BAB1 %>%
  tidyr::gather("predictor", "age", starts_with("age")) %>%
  ggplot(aes(x = age, y = ToT)) +
  facet_grid(predictor~.) +
  geom_point() +
  geom_smooth(se = F, method = "lm", fullrange = T)
```



By shifting the age variable, the whole data cloud is moved to the left. To see what happens on the inferential level, we repeat the LRM estimation with the two shifted variables:

```
M_age_shft <-
  stan_glm(ToT ~ 1 + age_shft, data = BAB1)

M_age_cntr <-
  stan_glm(ToT ~ 1 + age_cntr, data = BAB1)
```

We combine the posterior distributions into one multi-model posterior and read the *multi-model coefficient table*:

```
P_age <-
  bind_rows(posterior(M_age),
            posterior(M_age_shft),
            posterior(M_age_cntr))


T_age <- fixef(P_age)
T_age
```

Table 10: Estimates with 95% credibility limits

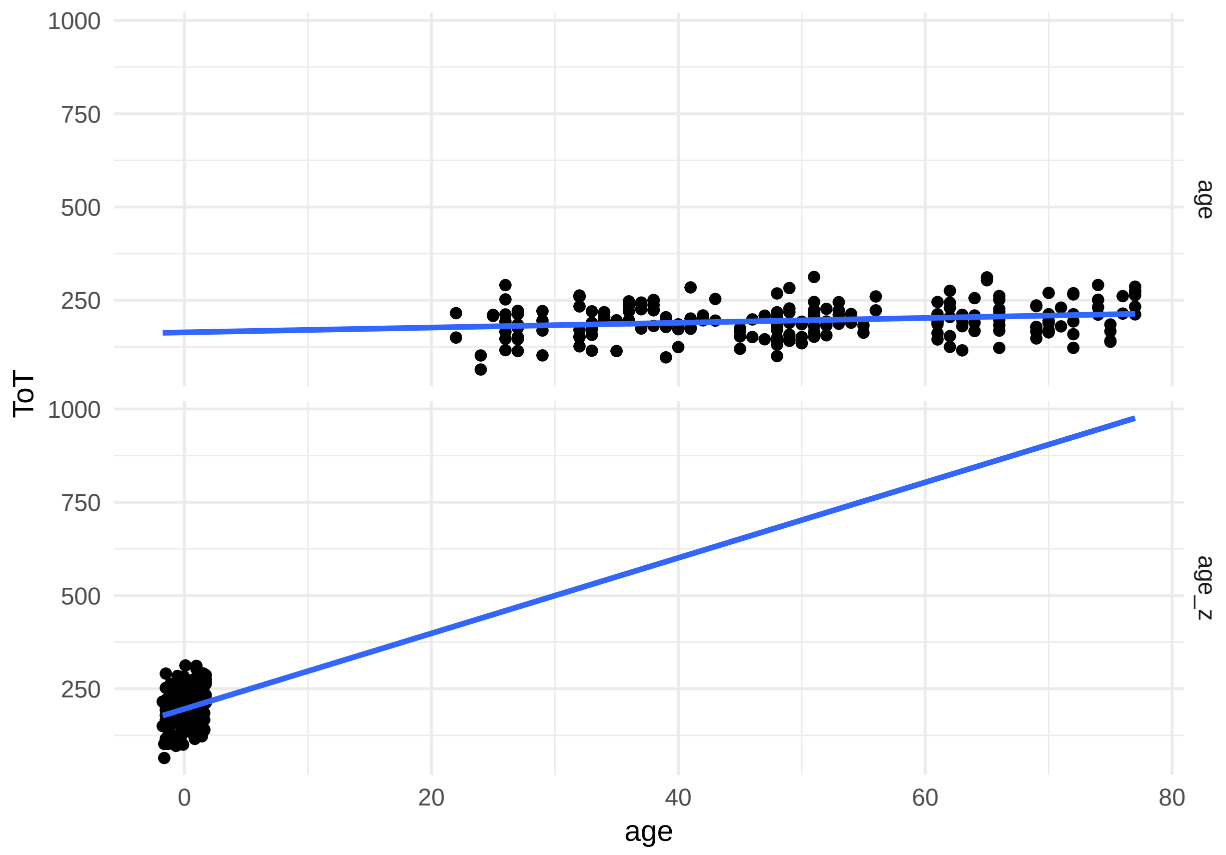
model	fixef	center	lower	upper
M_age	Intercept	164.090	143.152	185.18

model	fixef	center	lower	upper
M_age	age	0.642	0.234	1.04
M_age_cntr	Intercept	195.888	189.843	202.09
M_age_cntr	age_cntr	0.641	0.265	1.05
M_age_shft	Intercept	176.688	163.262	190.25
M_age_shft	age_shft	0.645	0.256	1.04

When comparing the regression results The shifted intercepts have moved to higher values, as expected. Surprisingly, the simple shift is not exactly 20 years. This is due to the high uncertainty of the first model, as well as the relation not being exactly linear (see Figure XY). The shifted age predictor has a slightly better uncertainty, but not by much. This is, because the region around the lowest age is scarcely populated with data for the very reason. Centering on the other hand results in a highly certain estimate, no surprisingly, as the region is densily populated. At the same time, the slope parameter practically does not change, neither in magnitude nor in certainty.

An even stronger standardization is *z-transformation*, where the predictor is centered at zero and all values are divided by the standard deviation, which results in a change of spread:

```
BAB1 <-
  BAB1 %>%
  mutate(age_z = (age - mean(age))/sd(age),
         age_cntr = age - mean(age))

BAB1 %>%
  tidyr::gather("predictor", "age", age, age_z) %>%
  ggplot(aes(x = age, y = ToT)) +
  facet_grid(predictor~.) +
  geom_point() +
  geom_smooth(se = F, method = "lm", fullrange = T)
```



A z-transformed variable is centered on zero and has a standard deviation of one. As can be seen, z-transformation has a considerable effect on the slope. Pulling the data cloud together pulls up the slope. Again, we run an LRM and compare against the original model:

```
M_age_z <-
  stan_glm(ToT ~ 1 + age_z, data = BAB1)
```

```
P_age <-  bind_rows(P_age, posterior(M_age_z))
T_age <- fixef(P_age)
T_age
```

Table 11: Estimates with 95% credibility limits

model	fixef	center	lower	upper
M_age	Intercept	164.090	143.152	185.18
M_age	age	0.642	0.234	1.04
M_age_cntr	Intercept	195.888	189.843	202.09
M_age_cntr	age_cntr	0.641	0.265	1.05
M_age_shft	Intercept	176.688	163.262	190.25
M_age_shft	age_shft	0.645	0.256	1.04
M_age_z	Intercept	195.873	189.852	202.37
M_age_z	age_z	10.118	3.720	16.24

As expected, the intercept matches that of the centered variable. The small deviations are due to the [random walk] (#random_walk) and disappear when running longer MCMC chains. The slope parameter has inflated dramatically. That, of course, is not a magical trick to obtain more impressive numbers, it is simply the effect of dividing the original variable by its standard deviation. A step of one on `age` is exactly one year, whereas in `age_z` it is *one standard deviation*. Z-transformation adjusts a variable by its observed dispersion, measured as standard deviations. In the case of age, this is not very desirable.

Years of age is a natural and commonly understood unit and my advice would be to use centering, leaving the unit size untouched. When dealing with variables that are pseudo-metric, rating scales in particular, z-transformation makes sense. Imagine you were stranded on a tropical island. While putting together a shelter, you realize of useful a meter stick is. The *ur meter* is thousands of kilometers away. Obviously, you can bootstrap yourself by picking up a reasoable straight stick and mark 10 (or eight) equally spaced sections. With z-transformation you are picking your unit of measurment on what you find, too. Still, "one standard deviation" is hard to swallow for anyone untrained. If we can assume the measurement to be normally distributed, which is frequently a good approximation for rating scale data, we can derive relative statements. As it happens, the central two standard deviations cover around two thirds of the full area, leaving one sixth to each tail. That allows to make quantile statements, such as: Within central two thirds regarding age, ToT moves up by $2\beta_1 = 10.12$ seconds, if age is truly Normal, of course.

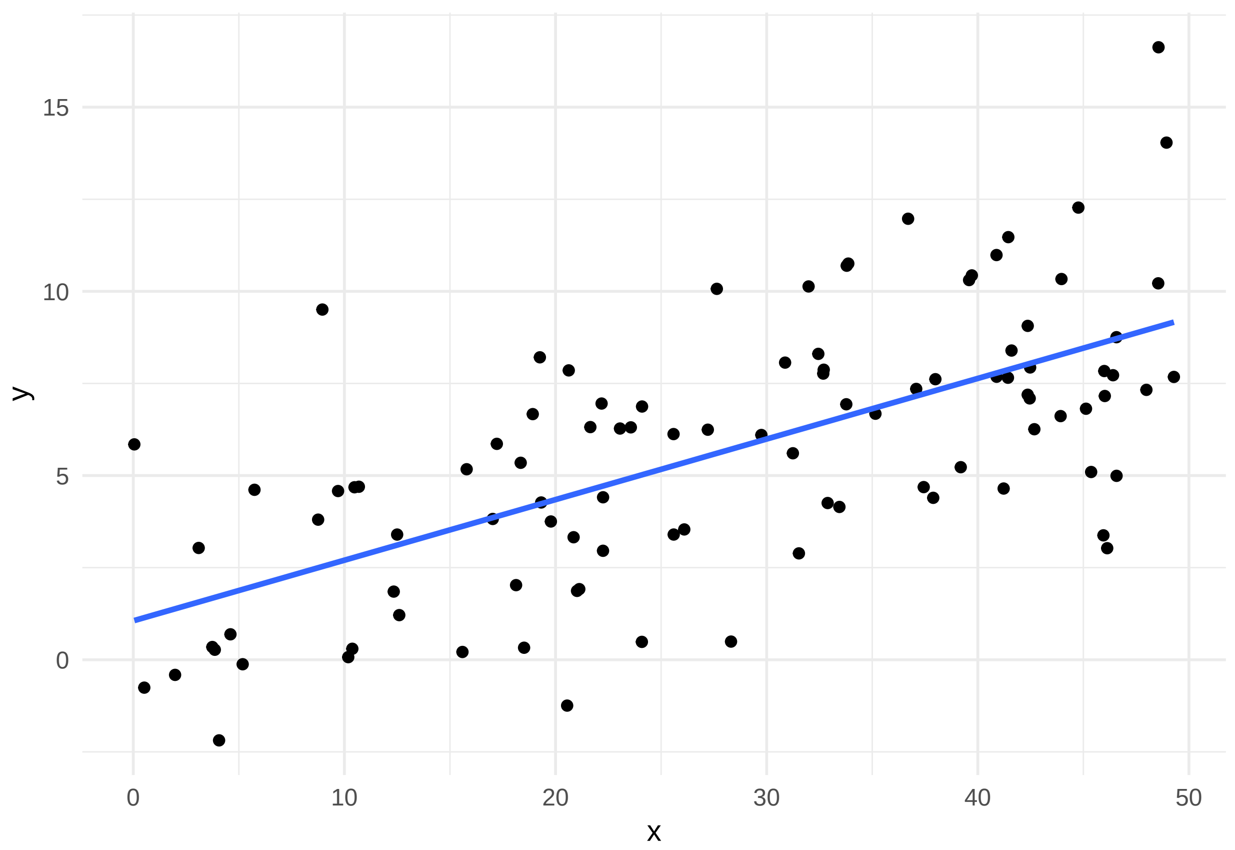
A general useful application of z-scores is to bootstrap a common unit for a diverse set of measures. Think of a battery of Likert scales, that all choose their own ranges. With z-scores we can compare the relative impact of several predictors in a population. In fact, that even makes sense for the age model. As it happens, the random variation parameter σ is given in standard deviation units, too. We can conclude that the age effect causes just a moderate fraction of all the variation observed. More research is needed.

Correlations

LRM render the relationship between two metric variables. A commonly known statistic that seems to do the same is Pearson's correlation coefficient r (@**??**#associations)). In the following, we will see that a tight connection between correlation and linear coefficients exists, albeit both having their own advantages. For a demonstration, we reproduce the steps on a smimulated data set where X and Y are linearly linked:

```
D_cor <-
  data_frame(x = runif(100, 0, 50),
             y = rnorm(100, x *.2, 3))
```

```
D_cor %>%
  ggplot(aes(x = x, y = y)) +
  geom_point() +
  geom_smooth(method = "lm", se = F)
```



Recall, that r is covariance standardized for dispersion and that a covariance is the mean squared deviance from the population mean. This is how the correlation coefficient is decontaminated from the idiosyncracies of the involved measures, their location and dispersion. In contrast, slope parameter in an LRM is a measure of association, too. It is agnostic of the overall location of measures, because this is captured by the intercept. However, dispersion remains intact. That makes that slope and intercept together retain information about location, dispersion and association of data and we can ultimately make predictions. Still, there is a tight re.lationship between Pearson r and a slope coefficient β_1, namely:

$$r = \beta_1 \frac{sd_X}{sd_Y}$$

For the sole purpose of demonstration, we here resort to the built-in non-Bayesian command `lm` for doing the regression.

```
M_cor <- lm(y ~ x, D_cor)
beta_1 <- stats::coef(M_cor)[2]

r <- beta_1 * sd(D_cor$x) / sd(D_cor$y)
r
```

```
##     x
## 0.649
```

The clue with Pearson r is that it normalized the slope coefficient by the variation found in the sample. This reminds of z-transformation as was introduced in @ref(transforming-variables). In fact, when both, predictor and outcome, are z-transformed before estimation, the coefficient equals Pearson's r, right away:

```
M_z <-
  D_cor %>%
  mutate(x_z = (x - mean(x))/sd(x),
         y_z = (y - mean(y))/sd(y)) %>%
  lm(y_z ~ x_z, .)

stats::coef(M_z)[2]
```

```
##   x_z
## 0.649
```

In regression modelling the use of coefficients prevales because they allow for predictions. However, correlation coefficients play a major role in two situations: in exploratory data analysis and in multilevel models. With the latter we will deal in @**??**re_correlations). For exploratory data analysis it is recommended to inspect pairwise correlations for the following reasons:

1. Correlations between predictors and responses are a quick and dirty assessment of the expected associations
2. Correlations between multiple response modalities (e.g., ToT and number of errors) indicate to what extent these responses can be considered exchangeable.
3. Correlations between predictors should be checked upfront to avoid problems arising from so-called colinearity.

Endlessly linear

On a deeper level the bizarre age = 0 prediction is an example of a principle, that will re-occur several times throughout this book.

In an endless universe, everything is finite.

A well understood fact about LRM is that they allow us to fit a straight line to data. A lesser regarded consequence from the mathematical underpinnings of such models is that this line extends infinitely in both directions. To fulfill this assumption, the outcome variable needs to have an infinite range, too, $y_i \in [-infty; \infty]$ (unless the slope is zero). Every scientifically trained person and many lay people know, that even elementary magnitude in physics are finite: all speeds are limited to $\approx 300.000 km/s$, the speed of light and temperature has a lower limit of -276°C (or 0K). If there can neither be endless acceleration nor cold, it would be daring to assume any psychological effect to be infinite in both directions.

The endlessly linear assumption (ELA) is a central piece of all LRM. From a formal perspective, the ELA is always violated in a universe like ours. So, should we never ever use a linear model and move on to non-linear models right away? Pragmatically, the LRM often is a reasonably effective approximation. From figure [G_eda_1] we have seen that the increase is not strictly linear, but follows a more complex curved pattern. This pattern might be of interest to someone studying the psychological causes of the decline in performance. For the applied design researcher it probably suffices to summarize the monotonous relationship by the slope coefficient. In @ref(line-by-line) we will estimate the age effects for designs A and B, separately, which lets us compare fairness towards older people.

As has been said theorists may desire more detailed picture and see disruptions of linearity as indicators for interesting psychological processes. An uncanny example of theoretical work will be given in polynomial regression. For the rest of us, linear regression is a pragmatic choice, as long as:

1. the pattern is monotonically increasing
2. any predictions stay in the observed range and avoid the boundary regions, or beyond.

Posterior predictions

In Bayesian regression models, *posterior predictions* are a simple, yet powerful concept to compare the fitted model with the original data. As [McElreath] points out, one should rather call them *retrodictions*, because they regard the real data one has, not the future. In yet another perspective they are *idealized responses* and as such address both, present and future. Posterior predictions are routinely compared to *observed values* y_i which are known beforehand, but still contain the randomness component. Fitting a linear model basically means separating the idealized effect from the randomness. When the model is somewhat well-specified, this succeeds but comes at the costs of uncertainty: the *predicted value is a random variable, too.*

In the BrowsingAB case, we have repeatedly recorded age of participant. LRM found the repeating pattern, that with every unit of age, ToT increased by 0.645 seconds. This pattern is what the model predicts, now and forever. If we ask: what is predicted for a person of age 45? We get the predicted value μ_i:

$$\mu_{age=45} = 176.688 + 0.645 * 45 = 205.707$$

The predicted value is our best guess under the LRM model. Like coefficients, it is uncertain to some degree, but we are setting this aside for the moment. What we know for sure is the *observed value* . A primitive procedure to get a best guess for someone of age 45 is to find one matching case in the data and take this as face value. The data set contains a few of those participants, but the situation is rather scattered and the prediction is not clear.

```
BAB1 %>%
  select(Part, age, ToT) %>%
  filter(age == 45) %>%
  kable()
```

Part	age	ToT
28	45	153
68	45	168
128	45	120
168	45	175

A more disciplined way to obtain predicted values is to use the standard command `predict` on the model object:

```
T_pred_age <-
  BAB1 %>%
  select(Part, age, ToT) %>%
  mutate(mu = predict(M_age_shft)$center)

T_pred_age %>%
  filter(age == 45) %>%
  kable()
```

Part	age	ToT	mu
28	45	153	193
68	45	168	191
128	45	120	194
168	45	175	194

We see that all participants of age 45 get the same prediction. What differs is the observed value, as this contains the influence of all random sources at the time of measuring. We will return to residuals in the next section.

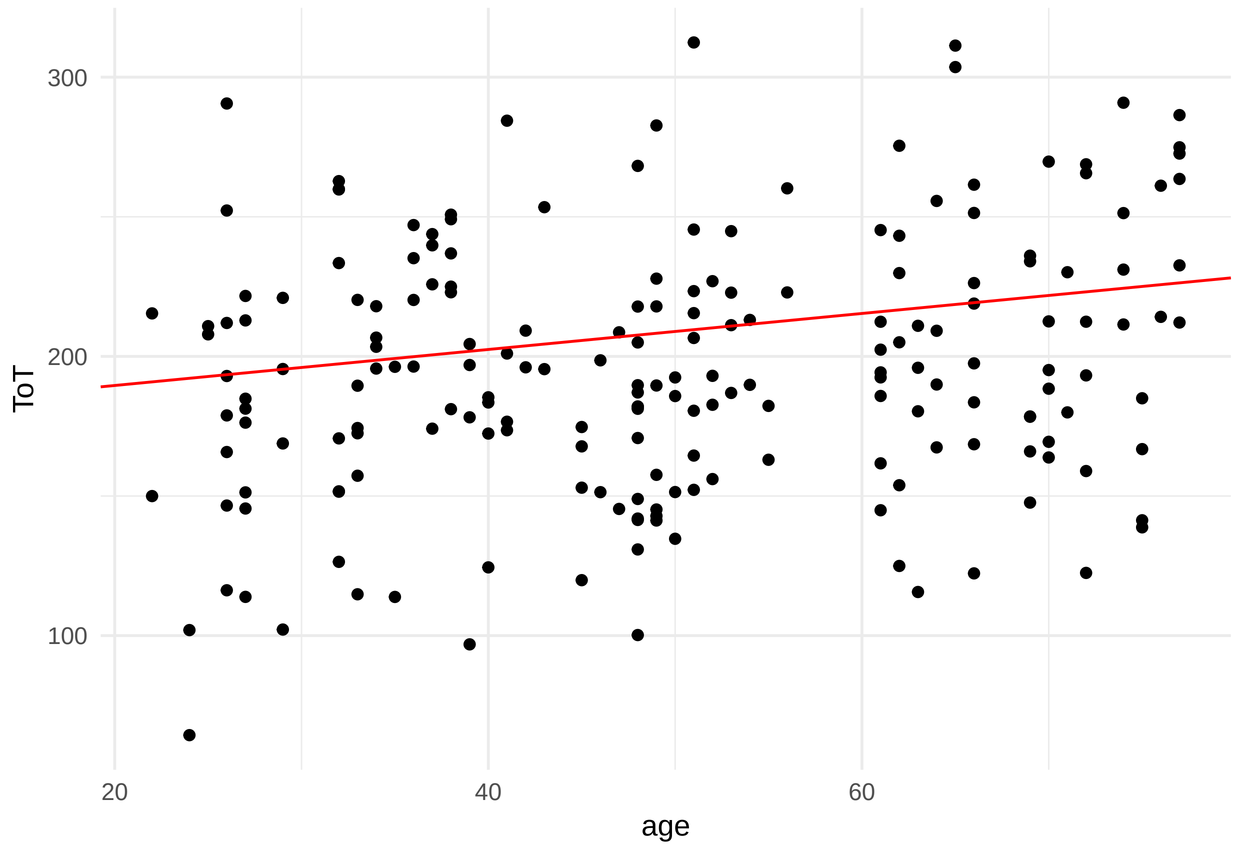
What if we wanted to get a best guess for an age that did not occur in our data set, say 43.5? Using the LR likelihood function above, we can estimate the expectation for any value we want. By entering the intercept and age estimates (`C_age`), we obtain:

```
C_age <- fixef(M_age_shft)$center
C_age[1] + 43.5 * C_age[2]
```

```
## [1] 205
```

With the same procedure, it is possible to plot observed and expected values next to each other:

```
G_pred_age <-
  BAB1 %>%
  ggplot(aes(x = age, y = ToT)) +
  geom_point() +
  geom_abline(intercept = C_age[1],
              slope = C_age[2],
              col = "red")

G_pred_age
```



```
detach(BrowsingAB)
```

This figure does not look too convincing. The regression line is rather flat, indicating a small effect of age. In addition, it looks like a lonely highway in a vast forest area. Just visually, a completely flat line or a slight negative slope would be equally credible.

- resid-predict plots as a general form to discover trends
- qq plots

Exercises

1. Examine the linear parameters of model `M_age_rtrn` and derive some impossible predictions, as was done in the previous section.

2. The BAB1 data set contains another outcome variable where the number of clicks was measured until the participant found the desired information. Specify a LR with age and examine the residual distribution. Is the Normality assumption reasonably defensible? What is the difference to home returns, despite both variables being counts?

3. Review the two figures in the first example of [GSR]. The observations are bi-modally distributed, nowhere near Gaussian. After (graphically) applying the model they are well-shaped. What does that tell you about checking residual assumptions before running the model?

A versus B: Comparison of groups

Another basic liknear model is the *comparison of groups* (CGM), which replaces the commonly known analysis of variance (ANOVA). In design research group comparisons are all over the place, for example:

- comparing designs: as we have seen in the A/B testing scenario
- comparing groups of people, like gender or whether they have a high school degree
- comparing situations, like whether someone uses an app on the go, or sitting still behind a computer

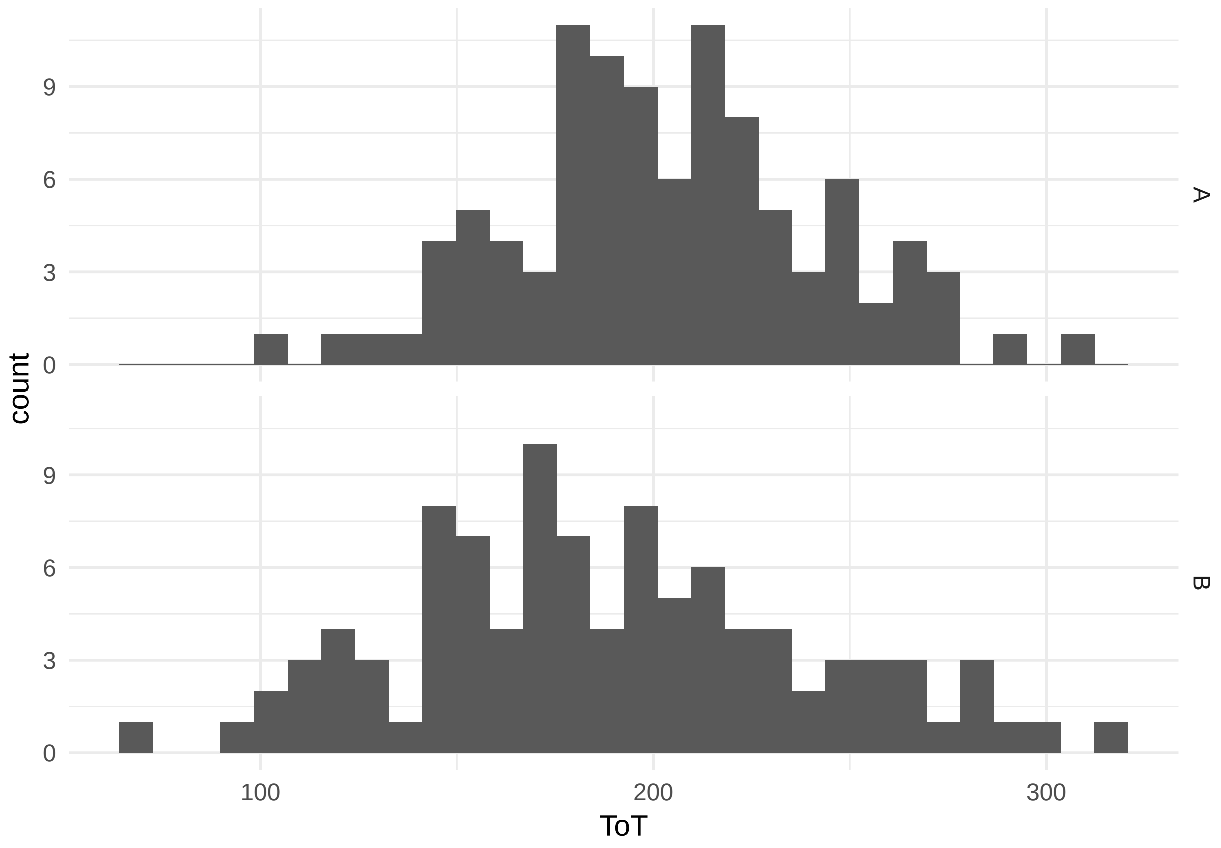
In order to perform a CGM, a variable is needed that establishes the groups. This is commonly called a *factor*. A factor is a variable that identifies members of groups, like "A" and "B" or "male" and "female". The groups are called *factor levels*. In the BrowsingAB case, the prominent factor is Design with its levels A and B.

Asking for differences between two (or more) designs is routine in design research. For example, it could occur during an overhaul of a municipal website. With the emerge of e-government, many municipal websites have grown wildly over a decade. What once was a lean (but not pretty) 1990 website over time has grown into a jungles, to the disadvantage for the users. The BrowsingAB case could represent the prototype of a novel web design, which is developed and tested via A/B testing at 200 users. Every user is given the same task, but sees only one of the two designs. The design team is interested in: *do two web designs A and B differ in user performance?*

Again, we first take a look at the raw data:

```
attach(BrowsingAB)
```

```
BAB1 %>%
  ggplot(aes(x = ToT)) +
  geom_histogram() +
    facet_grid(Design~.)
```



This doesn't look too striking. We might consider a slight advantage for design B, but the overlap is immense. We perform the CGM. Again, this is a two-step procedure:

1. the `stan_glm` command lets you specify a simple formula to express the dependency between predictors (education) and outcome variable (ToT). It performs the parameter estimation, using the method of *Markov-Chain Monte-Carlo Sampling*. The results are stored in a new object `M_Design`.
2. With the `fixef` command the estimates are extracted and interpreted

```
M_Design <-
  BAB1 %>%
  stan_glm(ToT ~ 1 + Design,
           data = .)
```

```
T_Design <- fixef(M_Design)
T_Design
```

Table 14: Estimates with 95% credibility limits

fixef	center	lower	upper
Intercept	203	194.7	212.21
DesignB	-15	-27.4	-3.05

```r
detach(BrowsingAB)
```

The model contains two parameters, the first one being the *intercept*, again. How can you have a "crossing point zero", when there is no line, but just two groups. You can't. When speaking of pure factor models, like the one above or the multifactorial models of the next section, the intercept has a different meaning, it is the *mean of a reference group*. Per default, stan_glm chooses the alphabetically first group label as the reference group, design A. We can therefore say that design A has an average performance of $203.36[194.7, 212.21]_{CI95}$.

The second parameter is the effect of "moving to design B". It is given as the *difference to the reference group*. With design B it took users $15.02[27.36, 3.05]_{CI95}$ less time to complete the task. This effect appears rather small and there is huge uncertainty about it. It barely justifies the effort to replace design A with B. If the BrowsingAB data set has some exciting stories to tell, the design difference is not it.

Not stupid: dummy variables

Are we missing something so far? We have not seen the model formula, yet. The CGM is a linear model, but this is not so apparent as it contains a factor, a non-metric variable. Linear model terms are a sum of products $\beta_i x_i$, but factors cannot just enter such a term. What would be the result of DesignB $\times \beta_1$?

Factors basically answer the question: *What group does the observation belong to?*. This is a label, not a number, and cannot enter the regression formula. *Dummy variables* solve the dilemma by converzing factor levels to numbers. A dumma variable represents only one level of a factor, by asking the simple question: *Does this observation belong to group DesignB?* The answer is yes/no and can be coded by a Boolean variable. For every level, a separate dummy variable is constructed to answer the simple yes/no question of membership. But, can Boolean variables enter arithmetic equations? At least, R assumes that this is safe, as they can always be converted to ones and zeros using `as.numeric`. Routinely, we can leave the whole dummy variable business to the linear model engine. Still, it is instructive to do it yourself once.

```r
attach(BrowsingAB)
```

```r
BAB1 <-  BAB1 %>%
  mutate(d_A = as.numeric(Design == "A"),
         d_B = as.numeric((Design == "B")))
BAB1 %>%
  select(Obs, Design, d_A, d_B, ToT) %>%
  sample_n(8) %>%
  kable()
```

Obs	Design	d_A	d_B	ToT
128	B	0	1	120
64	A	1	0	178
81	A	1	0	145
30	A	1	0	268
147	B	0	1	158
148	B	0	1	201
9	A	1	0	151
92	A	1	0	185

Vice versa, numbers in a logical context are always interpreted in complete reverse.

$$v = 1 \mapsto \text{TRUE} v = 0 \mapsto \text{FALSE}$$

Boolean variables can be used in all contexts, including numerical and categorical. It is unfortunate that we call them dummies in such a belittling manner, as they are truly bridges between the world of categories and arithmetic. They identify groups in our data set and switch on or off the effect in the linear term. For a factor G with levels A and B and zero/one-coded dummy variables `d_A` and `d_B`, the likelihood is written as:

$$\mu_i = d_{Ai}\beta_A + d_B\beta_{Bi}$$

When $d_{Ai} = 1, d_{Ai} = 0$, the parameter β_A is switched on, and β_B is switched off. An observation of group A gets the predicted value: $\mu_i = \beta_A$, vice versa for members of group B. All arithmetic commands in R do an implicit typecast when encountering a Boolean variable, e.g. `sum(TRUE, FALSE, TRUE)` (the result is 2). In contrast, regression engines interpret Boolean variables as categorical. Therefore, dummy variables have to be passed on as explicitly numeric to the regression engine. Only when a variables truly is zeroes and ones, it will be interpreted as desired. This has been done that with the explicit typecast `as.numeric` above.

Most research deals with estimating effects as differences and all regression engines quietly assume that what you want is a model with a reference group. When expanding the dummy variables manually, this automatism gets into the way and the needs to be switched off, explicitly. In Rs linear models formula language, that is done by adding the term `0 +` to the formula.

```
M_dummy <-
  stan_glm(ToT ~ 0 + d_A + d_B,
     data = BAB1)
```

```
fixef(M_dummy)
```

Table 16: Estimates with 95% credibility limits

fixef	center	lower	upper
d_A	203	194	213
d_B	188	179	197

In its predictions, the model `M_dummy` is equivalent to the fomer model `M_design`, but contains two effects that are exactly the group means. This model we call an *absolute group means model (AGM)*.

Regression engines expand dummy variables automatically, when they encounter a factor. However, when formally specifying the likelihood, dummy variables must be made explicit. When doing an AGM on a factor x_1 with $1, ..., k$ levels, the likelihood function becomes:

$$\mu_i = d_1\beta_{1[1]} + ... + d_k\beta_{1[k]}$$

In the remainder of the book, we are dealing with more complex models (e.g., multifactorial models in the next section), as well as factors with many levels (random effects in multi-level models @ref(multilevel-models)). With expanded dummy variables, the likelihood can become unduly long. However, many other textbooks avoid this altogether and the model is unambiguously specified in Rs regression formula language.

Up to this point, I have introduced dummy variables at the example of AGMs, where at any moment only one factor level is switched on. A more common CGMs would have a the following likelihood specification (with a factor of k levels):

$$\mu_i = \beta_0 + d_1\beta_{1[1]} + ... + d_k\beta_{1[k-1]}$$

For a factor with k levels in a CGM with intercept, $k-1$ dummy variables need to be constructed in the way shown above. As all non-reference levels are seen as difference towards the reference, *the reference level is set to always on*. To see so, we extract the dummy variables from the CGM on design with the standard command `model.matrix`. As you can see, for all observations the `(Intercept)` column takes the value `1`, whereas level `DesignB` is switched on and off.

```
BAB1 %>%
  select(Part, Design, ToT) %>%
  cbind(model.matrix(M_Design)) %>% ## <---
  as_data_frame() %>%
  sample_n(8) %>%
  kable()
```

Part	Design	ToT	(Intercept)	DesignB
85	A	178	1	0
104	B	181	1	1
76	A	213	1	0
132	B	203	1	1
127	B	114	1	1
196	B	205	1	1
199	B	291	1	1
167	B	211	1	1

Regression engines take care of factors, automatically, expanding the dummy variables under the hood. Still, by understanding the concept, we gained additional flexibility in specifying factorial models. One variant is to estimate an AGM, which leaves out the intercept. Of course, this can also be done right-away with the formula `ToT ~ 0 + Design`. Later, we will see a very practical application of AGMs, when drawing interaction plots. The second benefit is that we can specify the reference level as we want. It just depends on which dummy variable one sets to always-on. In the next section we will gain even more control over what the effects mean, by setting contrasts.

```
detach(BrowsingAB)
```

Getting it sharper with contrasts [TBC]

The default behavior of regression engines when encountering a factor is to select the first level as reference group and estimate all other levels relative to that. This fully makes sense when you are after the effect of a treatment and is therefore called *treatment contrasts*. Treatment contrasts do not have anything special to them. They are just a default of the regression engines, because so many people work experimentally.

Before we come to an understanding what contrasts are, let me point out what they are not: In @ref(dummy_variables) it was mentioned, that the AGM and CGM make exactly the same predictions (even though the formulas differ by the `+0` term). For contrasts, this holds in general. Setting contrasts never changes the predictions μ_i. In contrast, *contrast change what the coefficients β_i mean*. Recall, that in an CGM, β_1 has the meaning of "difference to the reference group", whereas in an AGM, it is "the mean of the second group".

In the following, I will illustrate contrasts that represent two different types of questions:

- To what extent does a group deviate from the overall mean in the sample?
- With three ordered factor levels, how large is the rolling effect, i.e. from level 1 to level 2, and the average of 1 and 2 to level 3?

So, how are contrasts set? Although, this is a bit odd, they are neither set by modifications to the regression formula, nor by additional command arguments. Instead, *contrasts are set as attributes of the factor variable*, which we can retrieve by and set by the standard command `contrasts`. For example, the variable Education has the three levels "Low", "Middle", "High", in that order:

```
attach(BrowsingAB)
```

```
levels(BAB1$Education)
```

```
## [1] "Low"    "Middle" "High"
```

```
contrasts(BAB1$Education) %>%
  kable()
```

	Middle	High
Low	0	0
Middle	1	0
High	0	1

If you believe to have seen something similar before, you are right. Contrast tables are related to dummy variables (@ref(dummy_variables)). The column Low is omitted in the contrasts table as it is the reference level with an always-on intercept. In the previous section I mentioned that you can choose the reference level freely by constructing the dummy variables accordingly. However, that would mean to always bypass the regression engines dummy handling. The package `forcats` provides a set of commands to change the order of levels. In the following code, the factor levels are reversed, such that High becomes the reference group:

```
BAB1 <-
  BAB1 %>%
  mutate(Education_rev = forcats::fct_rev(Education)) # <--
levels(BAB1$Education_rev)
```

```
## [1] "High"   "Middle" "Low"
```

```
contrasts(BAB1$Education_rev) %>%
  kable()
```

	Middle	Low
High	0	0
Middle	1	0
Low	0	1

```
detach(BrowsingAB)
```

If we were running a CGM on `Education_rev`, the intercept would represent level High, now. Again, this model would make the very same predictions and residuals. In that respect, it is the same model as before, only the meaning (and values) of the coefficients changes and become differences to the level High.

Sometimes, it is useful to have contrasts other than treatment, as this more closely matches the research question. A plethora of contrasts is known, I will introduce *deviation contrasts* and *successive difference contrasts* in the following.

[Example for deviation coding]

Successive difference coding (SDC) applies when one is interested in effects of progressive effects, such as performance gain in a number of sessions. Consider the following research situation. Shortly after the millennium, medical infusion pumps became infamous for killing people. Infusion pumps are rather simple devices that administer medication to a patients body in a controlled manner. Being widely used in surgery and intensive care, development of these devices must comply to national and international regulations. Unfortunately, the regulations of those days almost completely overlooked the human factor. While those devices would always function "as described in the user manual", they contained all kinds of severe violations of user-interface design rules, just to name few: foil switches with poor haptic feedback, flimsy alphanumeric LCD screenlets and a whole bunch of modes. Imagine a medical devices company has partnered with some renowned research institute to develop the infusion pump of the future. Users got interviewed and observed, guidelines were consulted, prototypes developed, tested and improved. At the end of the process the design was implemented as an interactive simulation. In the meantime, national agencies had reacted, too, and regulations now demand a routine user-oriented development cycle. One of the new rules says: "the design must undergo validation testing with trained users".

That means you have to first introduce and train your users to get fluent with the device, then test them. We [REF] thought that the training process itself is of immense importance. Why not test it, then? In the real study we tested everyone three times and traced individual progress. This requires a repeated measures analysis and we are not quite there, yet [see LMM].

[TODO:

- simulate IPump case, non-repeated
- demonstrate succ diff contr
- find example for deviation contrasts
- Contrasts extend the idea of dummy variables.
- Dummy variables become continuous, thereby more flexible in their effects

]

Sharper on the fly: derived quantities [TBD]

Contrasts are classic in aligning regression estimates with research questions that state a differences. With two groups A and B the following hypotheses of difference can be formed:

```
A - 0
B - 0
A - B
```

Exercises

1. The `simulate` function in case environment BrowsingAB lets you change the residual error (in standard deviations). Simulate three data sets with different residual variance, and estimate them by the same model. See how the uncertainty of effects behaves.
2. BrowsingAB, simulate several data sets of very small sample size. Observe how strongly the composition of education and age varies.
3. BAB1 contains the variable Education, which separates the participants by three education level (Low, Middle, High). Construct the dummy variables and run an AGM.
4. Specify the expanded CGM likelihood for education. Construct the dummy variables and run a regression.
5. Consider you wanted to use education level High as reference group. Create dummy variables accordingly and run the regression.

Putting it all together: multi predictor models

Design researchers are often forced to obtain their data under rather wild conditions. Users of municipal websites, consumer products, enterprise information systems and cars can be extremely diverse. At the same time, Designs vary in many attributes, affecting the user in many different ways. There are many variables in the game, and even more possible relations. With *multi predictor models* we can examine the simultaneous influence of everything we have recorded. First, we will see, how to use models with two or more metric covariates. Subsequently, we address the case of multi-factorial designs. Finally, we will see examples of models, where covariates and factors peacefully co-reside.

On surface: multiple regression models

Productivity software, like word processors, presentation and calculation software or graphics programs have evolved over decades. For every new release, dozens of developers have worked hard to make the handling more efficient and the user experience more pleasant. Consider a program for drawing illustrations: basic functionality, such as drawing lines, selecting objects, moving or colorize them, have practically always been there. A user wanting to draw six rectangles, painting them red and arranging them in a grid pattern, can readily do that using basic functionality. At a certain point of system evolution, it may have been recognized that this is what users routinely do: creating a grid of alike objects. With the basic functions this is rather repetitive and a new function was created, called "copy-and-arrange". Users may now create a single object, specify rows and columns of the grid and give it a run.

The new function saves time and leads to better results. Users should be very excited about the new feature, should they not? Not right so, as [Carroll in Rosson] made a very troubling observation: adding functionality for the good of efficiency may turn out ineffective in practice, as users have a strong tendency to stick with their old routines, ignoring new functionality right away. This troubling observation has been called the *active user paradox (AUP).*

Do all users behave that way? Or can we find users of certain traits that are involved. What type of person would be less likely to fall for the AUP? And how can we measure resistance towards the AUP? We did a study, where we explored the impact of two user traits *need-for-cognition (ncs)* and *geekism (gex)* on AUP resistance. To measure AUP resistance we observed users while they were doing drawing tasks. A moderately complex behavioral coding system was used to derive an individual AUP resistance score. Are users with high need-for-cognition and geekism more explorative and resistant to the AUP?

As we will see later, it is preferable to build a model with two simultaneous predictors. For instructive purposes we begin with two separate LRM, one for each predictor. Throughout the regression models we use z-transformed scores. Neither the personality nor the resistance scores have a natural interpretation, so nothing is lost in translation.

$$\mu_i = \beta_0 + \beta_{\text{ncs}} x_{\text{ncs}} \mu_i = \beta_0 + \beta_{\text{gex}} x_{\text{gex}}$$

```
attach(AUP)
M_1 <-
  AUP_1 %>%
  stan_glm(zresistance ~ zncs, data = .)

M_2 <-
  AUP_1 %>%
  stan_glm(zresistance ~ zgex, data = .)
detach(AUP)
```

@ref(tab:AUP_coef) shows the two separate effects (`M_1` and `M_2`). Due to the z-transformation of predictors, the intercepts are practically zero. Both personality scores seem to have a weakly positive impact on AUP resistance.

Next, we estimate a model that regards both predictors simultaneously. For linear models, that requires nothing more than to make a sum of all involved predictor terms (and the intercept). The result is a `multiple regression model (MRM)`:

$$\mu_i = \beta_0 + \beta_{\text{ncs}} x_{\text{ncs}} + \beta_{\text{gex}} x_{\text{gex}}$$

In Rs regression formula language, this likewise straight-forward. The `+` operator directly corresponds with the `+` in the likelihood formula.

```
attach(AUP)
M_3 <-
  AUP_1 %>%
  stan_glm(zresistance ~ zncs + zgex, data = .) #<--

detach(AUP)
```

For the comparison of the three models we make use of a feature of the package bayr: the posterior distributions of arbitrary models can be combined into one multi-model posterior object, by just stacking them upon each other. The coefficient table of such a multi-model posterior gains an additional column that identifies the model:

```
attach(AUP)

P <-
  bind_rows(posterior(M_1),
            posterior(M_2),
            posterior(M_3))

T_coef_3 <- P %>%
  posterior() %>%
  fixef()
T_coef_3
```

Table 20: Estimates with 95% credibility limits

model	fixef	center	lower	upper
M_1	Intercept	0.005	-0.308	0.306
M_1	zncs	0.377	0.054	0.699
M_2	Intercept	-0.004	-0.312	0.311
M_2	zgex	0.298	-0.029	0.605
M_3	Intercept	0.001	-0.306	0.317
M_3	zncs	0.300	-0.090	0.686
M_3	zgex	0.121	-0.273	0.508

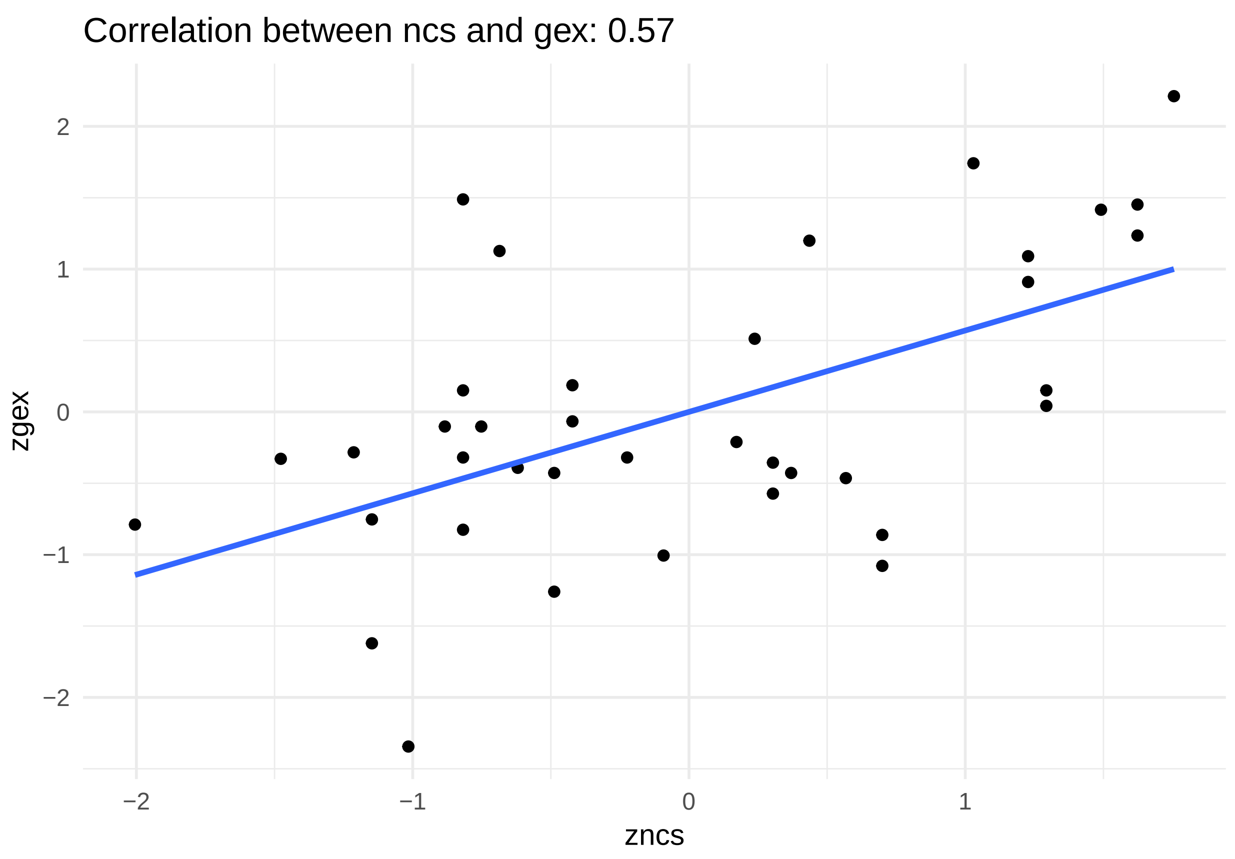
The intercepts of all three models are practically zero, which is a consequence of the z-transformation. Recall, that the intercept in an LRM is the predicted value, when the predictor is zero. In MRM this is just the same: here, the intercept is the predicted AUP resistance score, for when NCS and GEX are both zero.

When using the two predictors simultaneously, the overall positive tendency remains. However, we observe major and minor shifts: in the MRM, the strength of the geekism score is reduced to less than half: $0.12[-0.27, 0.51]_{CI95}$. NCS has shifted, too, but lost only little of its original strength: $0.3[-0.09, 0.69]_{CI95}$.

For any researcher who has carefully conceived research questions that appears to be a disappointing outcome. In fact, there is a reason for the loss of strength and ignoring this issue will mislead the interpretation of results.

The reason is that the two *predictors are correlated*. In this study, participants who are high on NCS also tend to have more pronounced geekism. @ref(AUP_corr_predictors) reveals the situation:

```r
G_eda_4 <-
  AUP_1 %>%
  ggplot(aes(x = zncs, y = zgex)) +
  geom_point() +
  geom_smooth(method = "lm", se = F) +
  labs(title = str_c("Correlation between ncs and gex: ",
              round(cor(AUP_1$zncs, AUP_1$zgex), 2)))

G_eda_4
```



```r
detach(AUP)
```

Participants with a higher NCS also tend to have more geekism. Is that surprising? Actually, it is not. People high on NCS love to think. Computers are a good choice for them, because these are complicated devices that make you think. (Many users may even agree that computers help you think, for example when analyzing your data with R.) In turn, geekism is a positive attitude towards working with computers in sophisticated ways, which means such people are more resistant towards the AUP.

[NCS: love to think] –> [GEX: love computers] –> [resist AUP]

When such a causal chain can be established, some researchers speak of a *mediating variable* GEX. Although a bit outdated [REF], *mediator analysis is correct when the causal direction of the three variables is known.* Then, a so-called step-wise regression is performed to find the pure effects. (A better alternative is structural equation modelling, SEM).

Unfortunately, in the situation here, the causal direction is partly ambiguous. We can exclude that the resistance test has influenced the personality scores, because of the order of apperance in the study. But, causally speaking, geekism may well preceed NCS. For example, computers reward you for thinking hard and, hence, you get used to it and make it your lifestyle. If you like thinking hard, then you probably also like the challenge that was given in the experiment.

[GEX: love computers] –> [NCS: love to think] –> [resist AUP]

In the current case, we can not distinguish between these two competing theories by data alone. This is a central problem in empirical research. An example, routinely re-iterated in social science methods courses is the observation that people who are more intelligent tend to consume more fresh vegetables. Do carrots make us smart? Perhaps, but it is equally plausible that eating carrots is what smart people do.

The issue is that a particular direction of causality can only be established, when all reverse directions can be excluded. Behavioural science reseachers know of only two ways to do so:

1. By the *arrow of time*, it is excluded that a later event caused a preceding one. In the AUP study, there is no doubt that filling out the two personality questionnaires cause the behaviour in the computer task, because of the temporal order.
2. In *strictly controlled experiments*, participants are assigned to the conditions, randomly. That has (virtually) happened in the BrowsingAB case, which gives it an unambiguous causal direction. This was not so, if participants had been allowed to choose themselves which design to test.

To come back to the AUP study: There is no way to establish a causal order of predictors. The correct procedure is to regard the predictors simultaneously (not step-wise), as in model `M_3`. This results in a redistribution of the overall covariance and the predictors are *mutually controlled.* In `M_2` the effect of GEX was promising at first, but then became spurious in the simultaneous model. Most of the strength was just borrowed from NCS by covariation. The model suggests that loving-to-think has a quite stronger association with AUP resistance than loving-computers.

That *may* suggest, but not prove, that geekism precedes NCS, as in a chain of causal effects, elements that are closer to the final outcome (AUP resistance), tend to exert more salient influence. But, without further theorizing and experimenting this is weak evidence of causal order.

The readers of a paper on geekism, NCS and the AUP will probably be more impressed by the (still moderate) effects in the separate models we had run, initially. The reason why one should not do that is that separate analyses suggest that the predictors are independent. To illustrate this at an extreme example, think of a study where users were asked to rate their agreement with an interface by the following two questions, before ToT is recorded:

1. The interface is beautiful
2. The interface has an aesthetic appearance?

Initial separate analysis shows strong effects for both predictors. Still, it would not make sense to give the report the title: "Beauty and aesthetics predict usability". Beauty and aesthetics are practically synonyms. For Gex and NCS this may be not so clear, but we cannot exclude the possibility that they are linked to a common factor, perhaps a third trait that makes people more explorative, no matter whether it be thoughts or computers.

So, what to do, when two predictors correlate strongly? First, we always report just a single model. Per default, this is the model with both predictors, simultaneously. The second possibility is to use a disciplined method of *model selection* @ref(model_selection) and remove this (or those) predictors that do not actually contribute to prediction. The third possibility is, that the results with both predictors become more interesting when including interaction effects @ref(interaction_effects)

Crossover: multifactorial models [TBC]

The only necessary precondition for statistical control is that you recorded the influencing variable. This has happened in the BrowsingAB study: the primary research question regarded the design difference, but the careful researcher also recorded the gender of participants.

What happened to the likelihood function when we moved from GMM to CGM and LRM? The effect of age was simply added to the intercept. For model on education level effects, we expanded the dummy variables and then added them all up. Indeed, the linear model is defined as a succession of linear terms $x_i\beta_i$ and nothing keeps us from adding further predictors to the model. Seeing is believing! The following code estimates a model with design and gender as predictors.

```r
attach(BrowsingAB)
```

```r
M_mpm_1 <-
  BAB1 %>%
  stan_glm(ToT ~ Design + Gender, data = .)
```

```r
T_fixef_mpm_1 <- fixef(M_mpm_1)
T_fixef_mpm_1
```

Table 21: Estimates with 95% credibility limits

fixef	center	lower	upper
Intercept	203.349	191.9	215.10
DesignB	-15.050	-27.8	-2.68
GenderM	0.196	-12.8	12.71

By adding gender to the model, both effects are estimated simultaneously. In a *factorial MPM* the intercept is a reference group, too. Consider that both factors have two levels, forming a $2x2$ design with groups: A-F, A-M, B-F, B-M. The first one, A-F, has been set as reference group. Women in condition A have an average ToT of $203.35[191.94, 215.1]_{CI95}$ seconds. The other two fixed effects are, once again, differences to the reference. Here, nor does gender do much to performance, nor does the design effect really change, compared to the CGM.

[likelihood][interaction plot]

Line-by-line: regression in groups [TBC]

Recall that dummy variables make factors compatible with linear regression. No barriers are left for combining factors and covariates in one model. For example, we can estimate the effects age and design simultaneously:

```r
M_mpm_2 <-
  BAB1 %>%
  stan_glm(ToT ~ Design + age_shft, data = .)
```

```r
T_fixef_mpm_2 <-  fixef(M_mpm_2)
T_fixef_mpm_2
```

Table 22: Estimates with 95% credibility limits

fixef	center	lower	upper
Intercept	184.208	169.940	198.79
DesignB	-14.981	-27.804	-2.14
age_shft	0.648	0.254	1.03

```r
detach(BrowsingAB)
```

Once again, we get an intercept, first. Recall, that in LRM the intercept is the the performance of a 20-year old (age shifted). In GCM it was the mean of the reference group. When *marrying factors with a covariates, the intercept is point zero in the reference group.* The predicted average performance of 20-year old with design A is $184.21[169.94, 198.79]_{CI95}$. The age effect has the usual meaning: by year of life, participants get $0.65[0.25, 1.03]_{CI95}$ seconds slower. The *factorial effect* B is a *vertical shift of the intercept.* 20-year old in condition B are $14.98[27.8, 2.14]_{CI95}$ seconds faster. In fact, this holds for all ages, as can be seen in the following figure. The model implies that the age affect is the same with both designs, which is not true, as we will see later.

[likelihood][interaction plot]

Residual analysis

Assessing predictive power

```
attach(BrowsingAB)
```

When residuals are pronounced, predictions are inaccurate, which is undesireable. A model that reduces the residuals, may provide better predictions. In the current case, we may wonder: does the age predictor effectively reduce the residual standard error σ_ϵ as compared to the GMM? We fit the GMM for comparison purposes and compare the standard error. This does not look convincing, as the reduction is marginal.

```
M_0 <-
  BAB1 %>%
  stan_glm(ToT ~ 1, data = .)
```

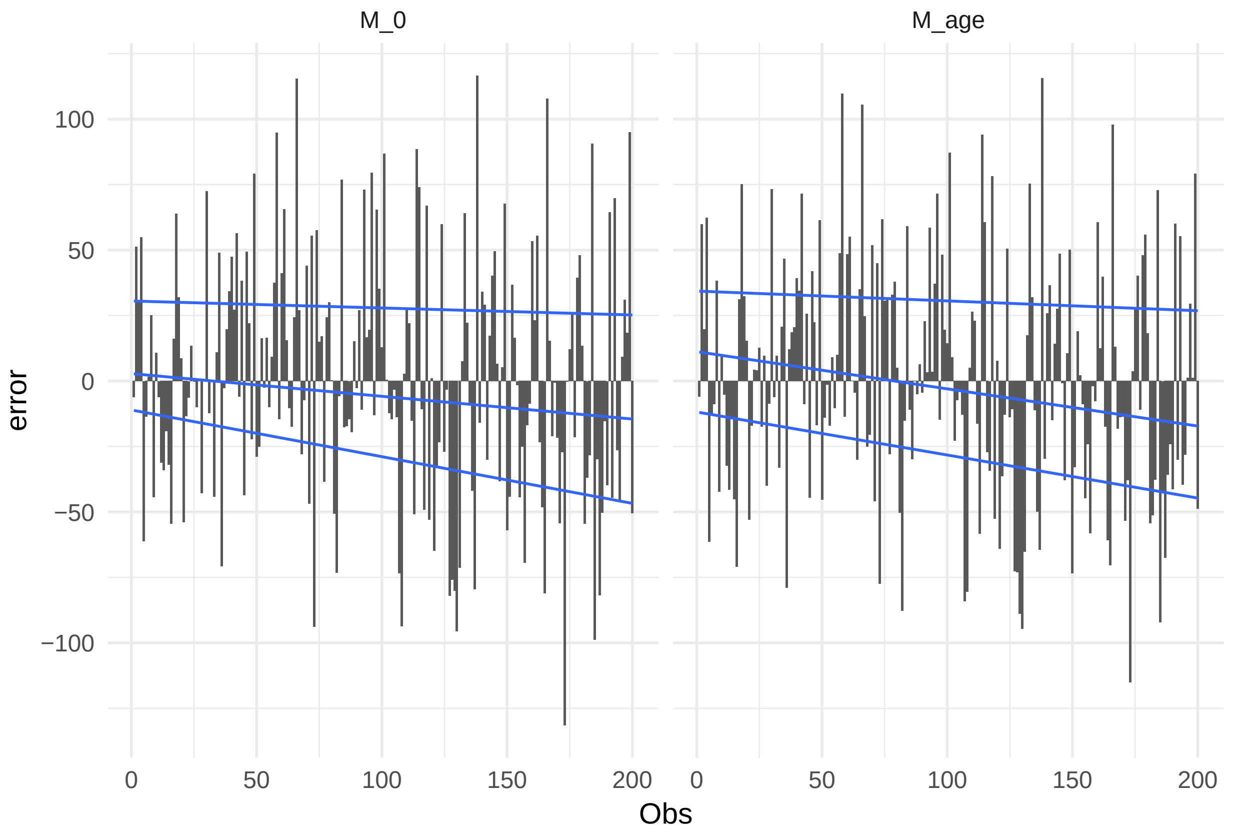
```
bind_rows(
  posterior(M_0),
  posterior(M_age_cntr)) %>%
  coef(type = "disp")
```

Table 23: Estimates with 95% credibility limits

model	center	lower	upper
M_0	46.1	42.0	51.2
M_age_cntr	45.2	41.1	49.9

Does the age variable really have that little predictive value? Seeing is believing. The following graph shows the individual residuals for both models as a column plot. The two caterpillars have about the same overall width and there is no noticable reduction in residual magnitude by adding the age predictor.

```
T_resid <-
  BAB1 %>%
  select(Obs, age) %>%
  mutate(
    M_0 = residuals(M_0),
    M_age = residuals(M_age))

G_resid_age_1 <-
  T_resid %>%
  mutate(Obs_ordered = min_rank(age)) %>%
  gather(model, error, -Obs, -Obs_ordered, -age) %>%
  ggplot(aes(x = Obs, y = error)) +
  facet_grid(~model) +
  geom_col(position = "dodge") +
  geom_quantile()
G_resid_age_1
```



When adding a predictor to a model, the least one would expect is a noticable reduction in error and @ref(BAB1_resid_age_1) confirms that the age predictor is pretty useless.

```
detach(BrowsingAB)
```

Normal distribution

So far, we have seen two linear models, the GMM and the LRM. They only differ in how they sketch the relation between predictor variables and predicted values. In the remainder of this chapter on linear models, we will encounter a few more building blocks for the likelihood part and these will allow us to specify a wealth of complex models. However, the random term will stoically stay the same:

$$y_i \sim N(\mu_i, \sigma_\epsilon)$$

In words, the random term says: *observed values y_i are drawn from a Normal distribution with the predicted value μ_i as mean and a fixed standard deviation σ_ϵ.* As we have seen in @ref(distributions), Normal distributions are one pattern of randomness among many and this choice may therefore be appropriate, or not. One heuristic that may justify this choice is that the observed values are located rather in the center of the scale of measurement. That is certainly much better, than to blindly stick to the Normal random pattern just for convenience. Even better is to check the assumption of Normally distributed randomness. In the following, we will examine the underlying assumptions closer and apply graphical techniques for verification. Before we delve into more depth, I would like to contrast the overall tenor in this section (and [MODSEL]) to the workflow frequently encountered in classic statistics. Boldly speaken, classicically trained researchers often sem to imply, that such assumptions needed to be checked beforehand. In the process called *assumption checking*, arcane non parametric tests are carried out, before the researcher actually dares to hit the button labelled as *RUN ANOVA*. As we will see now, the order of actions is just the other way round. In the workflow called *model criticism*, we start by contemplating what may be a reasonable model, using heuristics or even past data. Then the model is immediatly executed and the estimated model itself undergoes a routine checkup. In linear models, verifying the random pattern assumptions grounds on extracting the estimated residuals from the model object and is hence called *residual analysis*.

In the notation displayed above, there are possibly as many distributions as there are observed values (due the subscript in μ_i). It appears impossible to evaluate not just one distribution but such many. However, an equivalent notation is routinely used for linear models, that specifies just one *residual distribution*. For the LRM that is:

$$\mu_i = \beta_0 + \beta_1 x_1 y_i = \mu_i + \epsilon_i \epsilon_i \sim N(0, \sigma_\epsilon)$$

In this notation, observed values y_i are decomposed into predicted values and *individual* residuals ϵ_i. These are frequently called *errors*, hence the greek symbol ϵ. The standard deviation of residuals σ_ϵ is commonly called the *standard error*. The random pattern of the model can now be expressed as a single Normal distribution. The reason why I do not use this notation routinely, is that it only works for linear models, but not for models with other random patterns. More specifically, Generalized Linear Models @ref(generalized_linear_models) cannot be specified that way. But, for the purpose of residual analysis, it appears more intuitive.

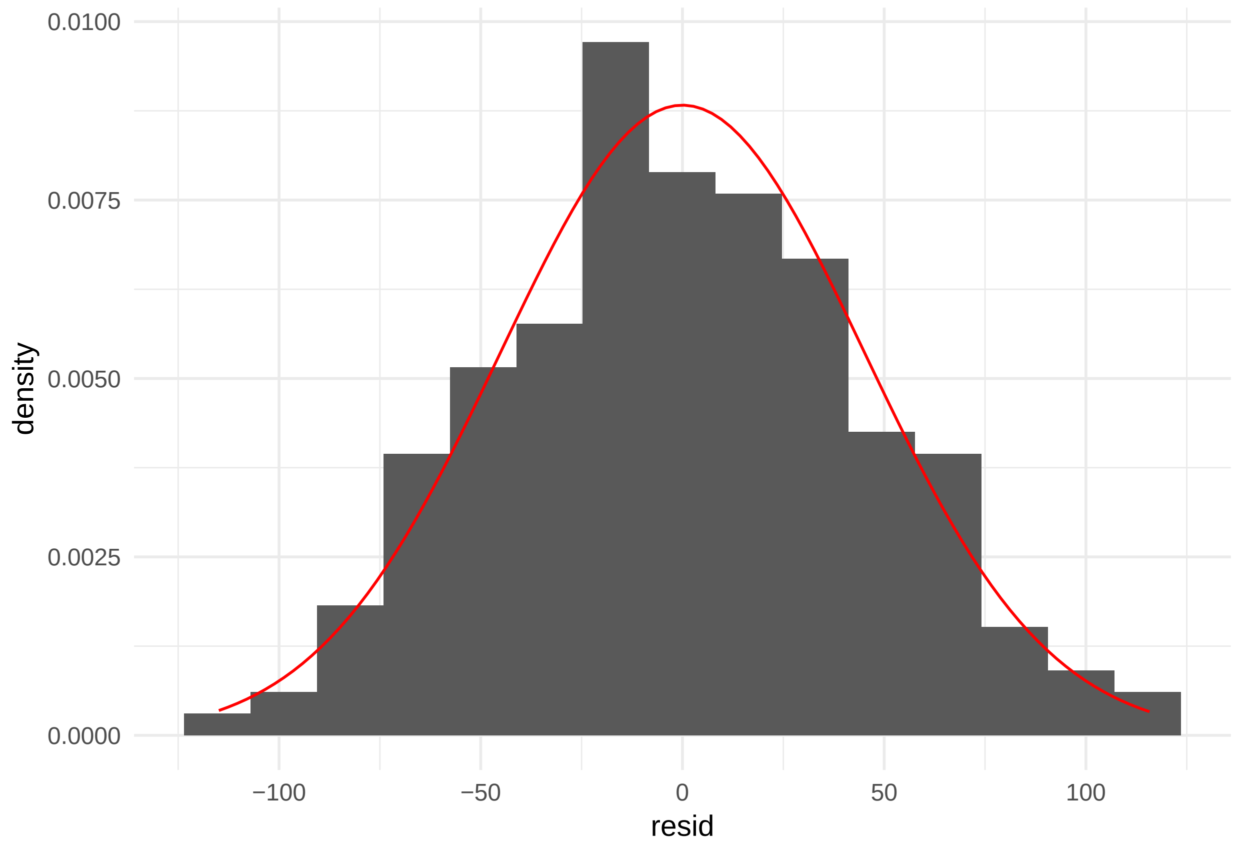
The first assumption of randomness underlying the linear model simply is that the distribution follows Normal distribution. Visually, Normal distributions is characterized by:

- one curved peak (unimodality)
- from which density smoothly declines towards both ends (smoothness)
- at same rates (symmetry)

For a rough evaluation of this assumption, it suffices to extract the residuals from the model at hand and plot it as a distribution. The `residuals` command returns a vector of residual values, exactly one per observation. With the vector of residuals at hand, we can evaluate this assumption by comparing the residual distribution to its theoretical form, a perfect bell curve. The following command chain extracts the residuals from the model and pipes them into the ggplot engine to create a histogram. With `stat_function` an overlay is created with the theoretical Normal distribution, which is centered at zero. The standard error σ_ϵ has been estimated alongside the coefficients and is extracted using the function `bayr::coef`.

```r
attach(BrowsingAB)
```

```r
G_resid_age_shft <-
  data.frame(resid = residuals(M_age_shft)) %>%
  ggplot(aes(x = resid)) +
  geom_histogram(aes(y = ..density..), bins = 15) +
  stat_function(fun = dnorm,
                args = c(mean = 0,
                         sd = coef(M_age_shft, type = "disp")$center),
                colour = "red")

G_resid_age_shft
```



The match of residual distribution with the theoretical distribution is not perfect, but overall this model seems to sufficiently satisfy the Normality assumption. To give a counter example, we estimate the same model using the outcome variable `returns`, which captures the number of times a participant had (desparately) returned to the homepage.

```
M_age_rtrn <-
  stan_glm(returns ~ 1 + age_shft, data = BAB1)
P_age_rtrn <- posterior(M_age_rtrn)
```

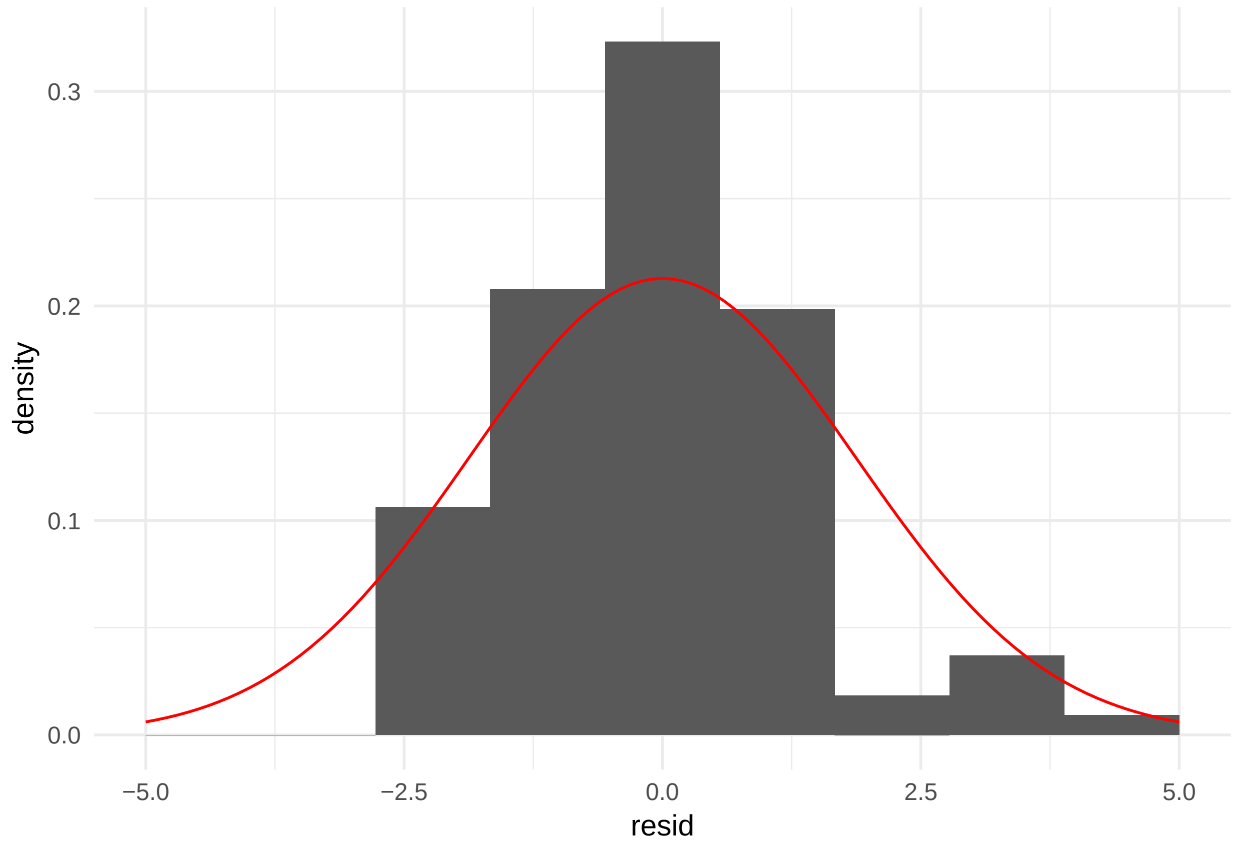
```
T_age_rtrn <- coef(P_age_rtrn)
T_age_rtrn
```

Table 24: Estimates with 95% credibility limits

parameter	type	fixef	center	lower	upper
Intercept	fixef	Intercept	2.563	1.99	3.099
age_shft	fixef	age_shft	-0.004	-0.02	0.013
sigma_resid	disp	NA	1.875	1.71	2.073

```
C_age_rtrn_disp <-
  T_age_rtrn %>%
  filter(type == "disp") %>%
  select(center) %>%
  as.numeric()
```

```
G_resid_age_rtrn <-
  data_frame(resid = residuals(M_age_rtrn)) %>%
  ggplot(aes(x = resid)) +
  geom_histogram(aes(y = ..density..), bins = 10) +
  stat_function(fun = dnorm,
                args = c(mean = 0,
                         sd = C_age_rtrn_disp),
                colour = "red") +
  xlim(-5, 5)
G_resid_age_rtrn
```



```
detach(BrowsingAB)
```

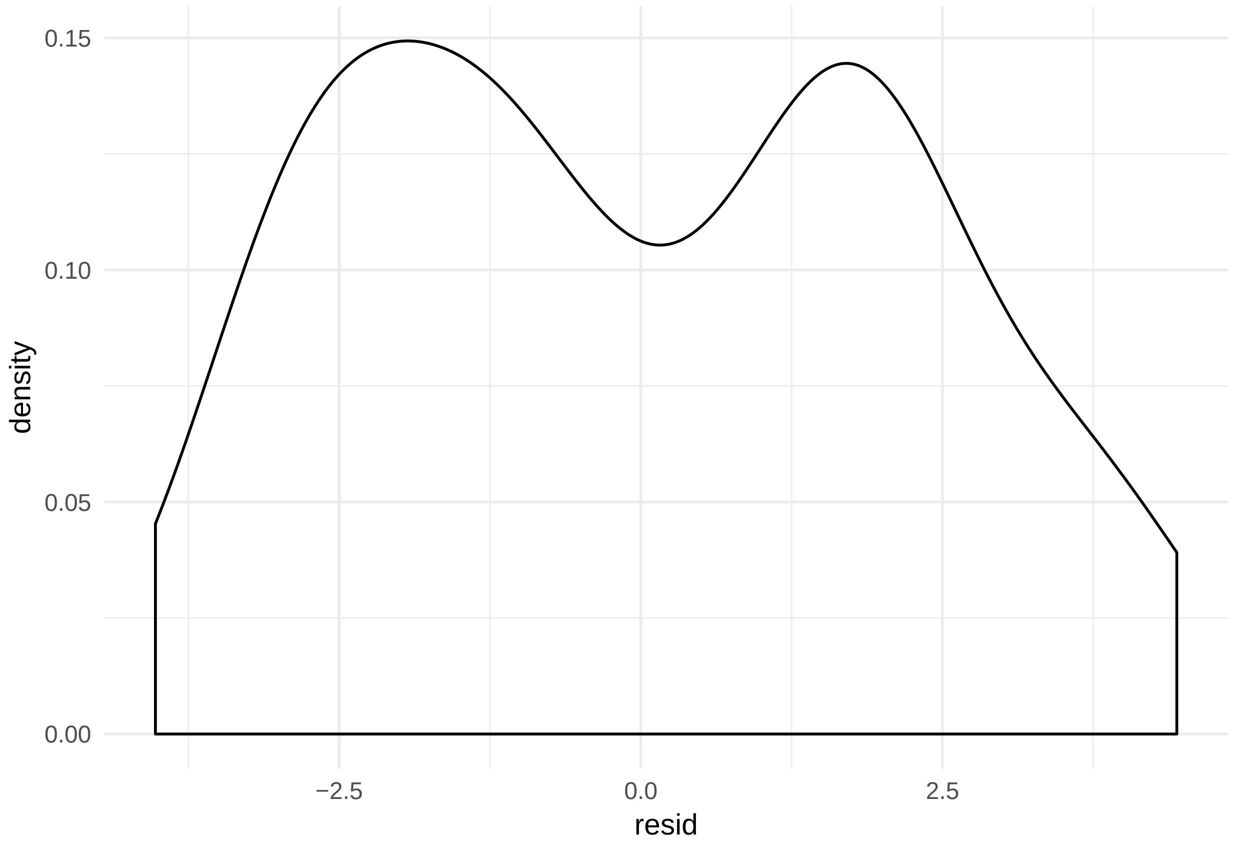
The estimation produces the usual coefficients, as well as a standard error. However, the residuals not even remotely resemble the theoretical curve. While it is unimodal, it appears rather asymmetric, with a steep rise to the left and a long tail to the right. That is a typical outcome when count measures get too close to the left boundary. How about unimodality? We have not discussed any multimodal theoretical distributions in @ref(distributions), but one has been displayed in @ref(first_program). In brief, a bimodal residual distribution can arise, when two groups exist in the data, which lay far apart. The following code illustrates the situation by simulating a simple data set with two groups, that is fed into a GMM.

```
attach(Chapter_LM)
```

```r
set.seed(42)
D_bimod <-
  bind_rows(
    data_frame(Group = "A", y = rnorm(50, 4, 1)),
    data_frame(Group = "B", y = rnorm(50, 8, 1))
  )
```

```r
M_bimod <- stan_glm(y ~ 1, data = D_bimod, iter = 200)
```

```r
D_bimod  %>%
  mutate(resid = residuals(M_1)) %>%
  ggplot(aes(x = resid)) +
  geom_density()
```



These two deviations from Normal distribution have very different causes: asymmetry is caused by scales with boundaries. This is an often arising situation and it is gracefully solved by Generalized Linear Models @ref(GLM). This is a family of models, where each member covers a certain type of measurement scale. In the above example, Poisson regression, applies, taking care of count measures. Mulitmodality is caused by heterogeneous groups in the data, like experimental conditions, design or type of user. For a grouping structure to cause distinguished multimodality, differences between groups have to be pronounced, in relation to the standard error. It is often the case, that these variables are controlled conditions, such as in an AB test. It is also quite likely that strong grouping structures can be thought of beforehand and be recorded. For example, in usability tests with diverse user samples, in almost comes natural to distinguiish between users who have used the design before, and those who didn't. If the grouping variable is recorded, the solution is group comparison models @ref(CGM), soon to be introduced.

Visual assessment of symmetry and unimodality is simple and effective in many cases. But, Normal distributions are not the only to have these properties. At least logistic distributions and t distributions have these, too, with subtle different in curvature. Normal and t distributions differ in how quickly probability drops in the tails. Normal distributions drop much faster such that extreme events are practically impossible. With t-distributions, extreme values drop in probability, too, but the possibility of catastrophies (or wonders) stays substantial for a long time.

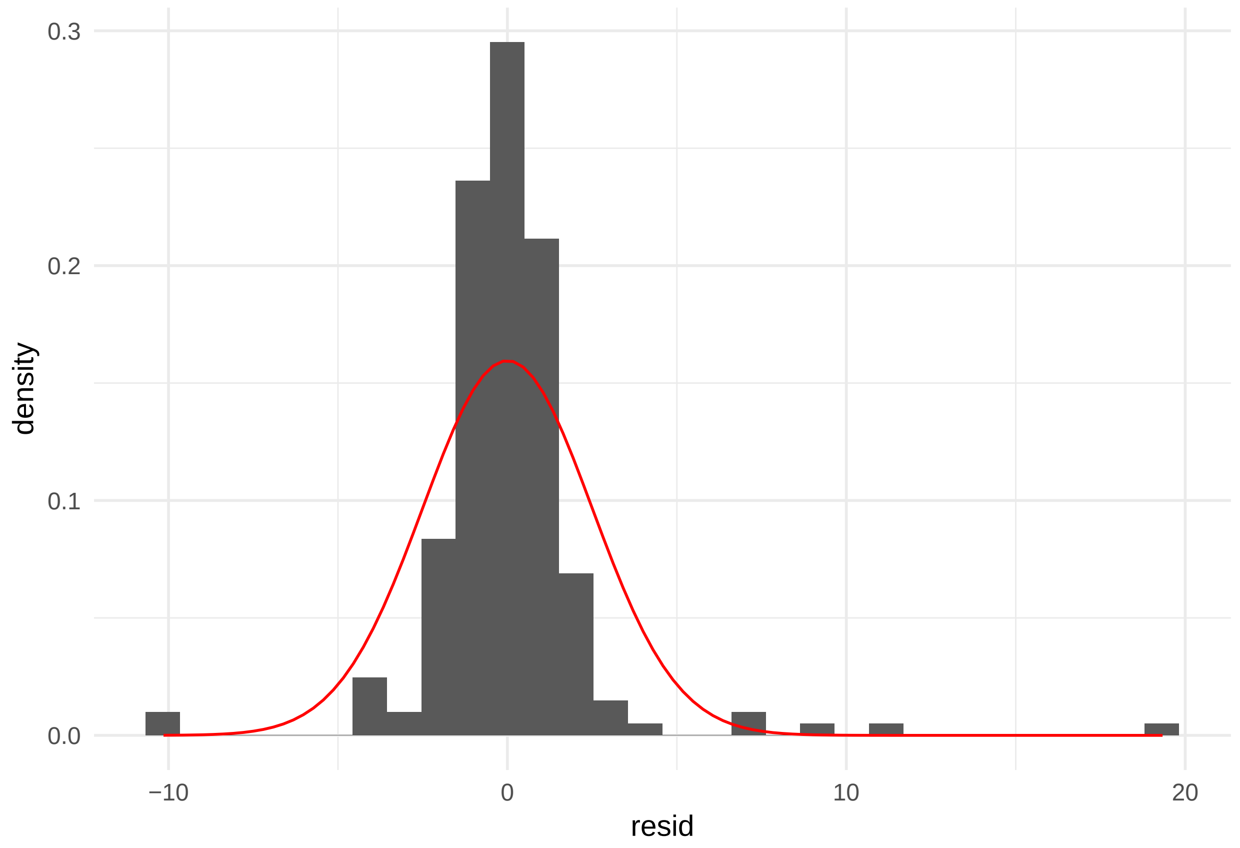
Provided one has a small abundance of data, *quantile-quantile (qq) plots* can be used to evaluate subtle deviations in curvature (and symmetry and unimodality). In qq plots, the observed and theoretical distributions are both flattened and put against each other. This is a powerful and concise method, but it is harder to grasp. The following code illustrates the construction of a qq-plot that compares GMM residuals of a t-distributed measure against the Normal distribution. We simulate t-distributed data, run a GMM and extract residuals, as well as the standard error σ_ϵ.

```r
set.seed(2)
D_t <- data_frame(y = rt(200, 2))
```

```r
M_t <- stan_glm(y ~1, data = D_t, iter = 200)
```

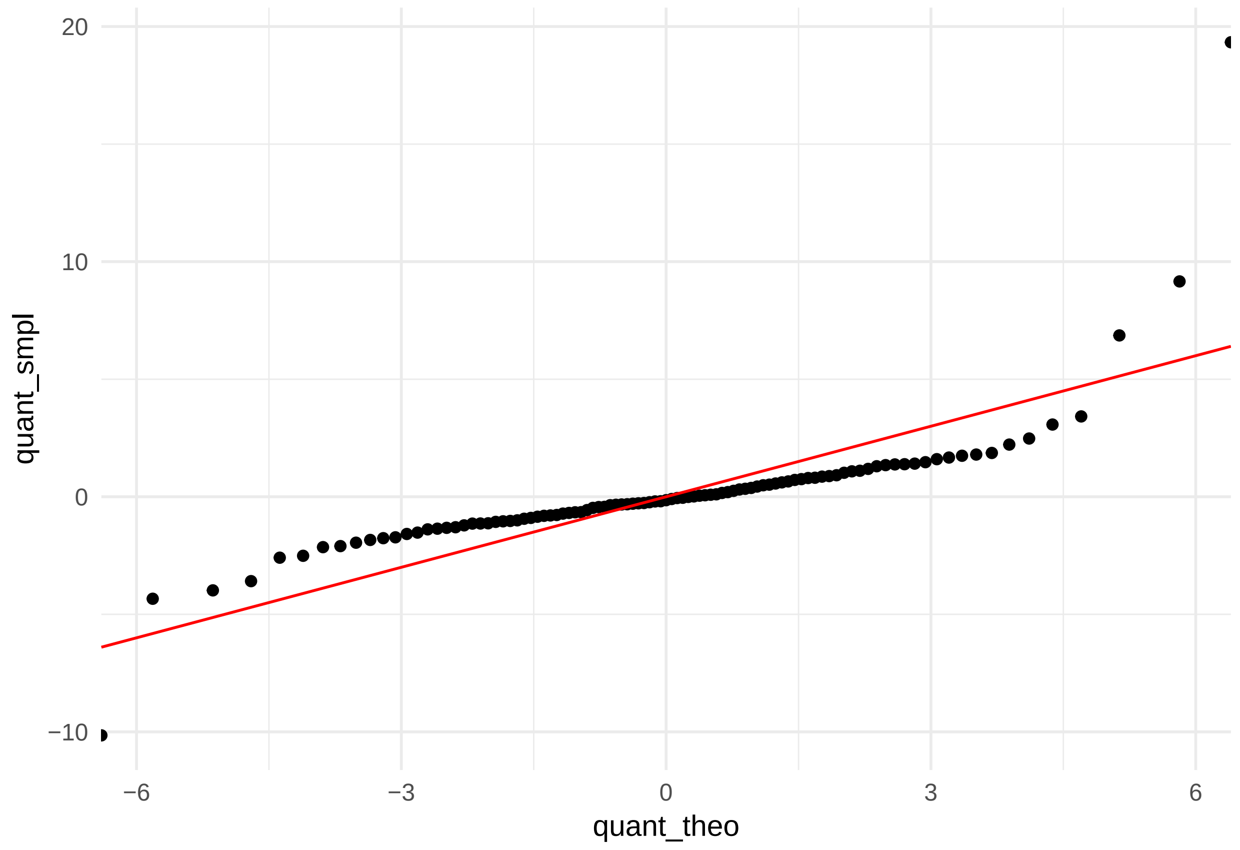
We obtain the following residual and theoretical distributions. It is approximately symmetric and unimodal, but the curvature seems to be a bit off.

```r
D_t <-   mutate(D_t, resid = residuals(M_t))

C_sigma <- rstanarm::sigma(M_t)

D_t %>%
  ggplot(aes(x = resid)) +
  geom_histogram(aes(y = ..density..)) +
  stat_function(fun = dnorm,
                args = c(mean = 0,
                         sd = C_sigma),
                colour = "red")
```



The next step is where the two curves get flattened. First, we compute a sequence of quantiles with fixed steps, say 1%, 2%, ... 99%%. Finally, theoretical and observed quantiles are fed into a scatterplot.

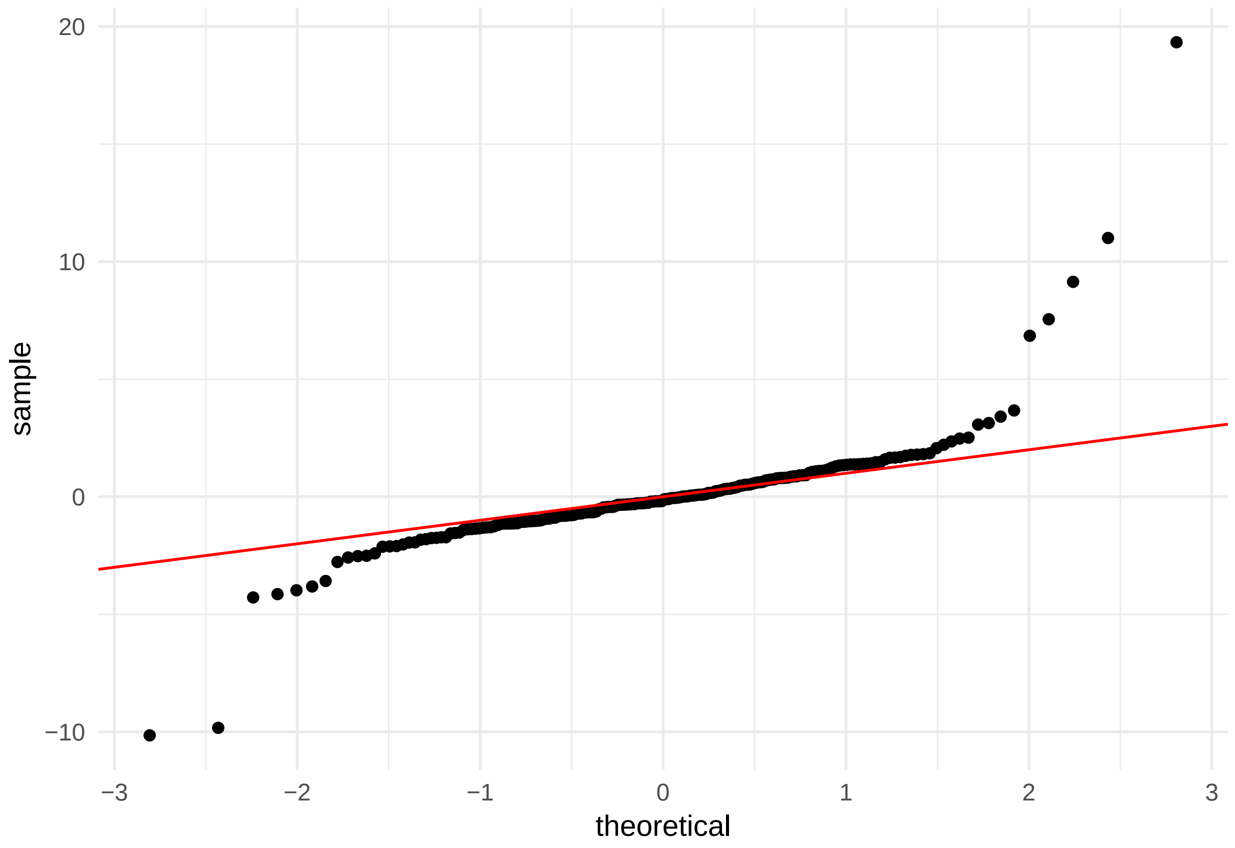
```
D_QQ <- data_frame(step = 0:100/100,
           quant_smpl = quantile(D_t$resid, step),
           quant_theo = qnorm(step, 0, C_sigma))

D_QQ %>%
  ggplot(aes(x = quant_theo, y = quant_smpl)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0, col = "red")
```



In the ideal case, they match perfectly, and the quantiles are on a straight line. Instead, we see a rotated sigmoid shape and this is typical for fat-tailed distributions such as t. The shape is symmetric with turning points at around -4 and 4 on the theoretical scale. In the middle part the relation is almost linear, however, not matching a 1-by-1. The t distribution loses probability mass rather quickly when moving from the center to the tuzrning points. here. From these points on the theoretical quantiles start lag behind. The lower and upper 1% sampled quantiles go to much more extreme values, ranging from -10 to almost 20, whereas the Normal distribution renders such events practically impossible. Generally, a rotated sigmoid shape is typical for fat tailed distributions. The problem of misusing a Normal distribution is that it dramatically underestimates extreme events. Have you ever asked yourself, why in the 1990s, the risk for a nuclear meltdown were estimated to be one in 10.000 years, in face of two such tragic events in the past 40 years? Perhaps, researchers used the Normal distribution for the risk models, under-estimating the risk of extreme events.

The ggplot engine provides an easy to use geometry for qqlots, which lets us further explore deviant patterns. Variables with t distribution take an inverse-sigmoid shape due to their fat tails.

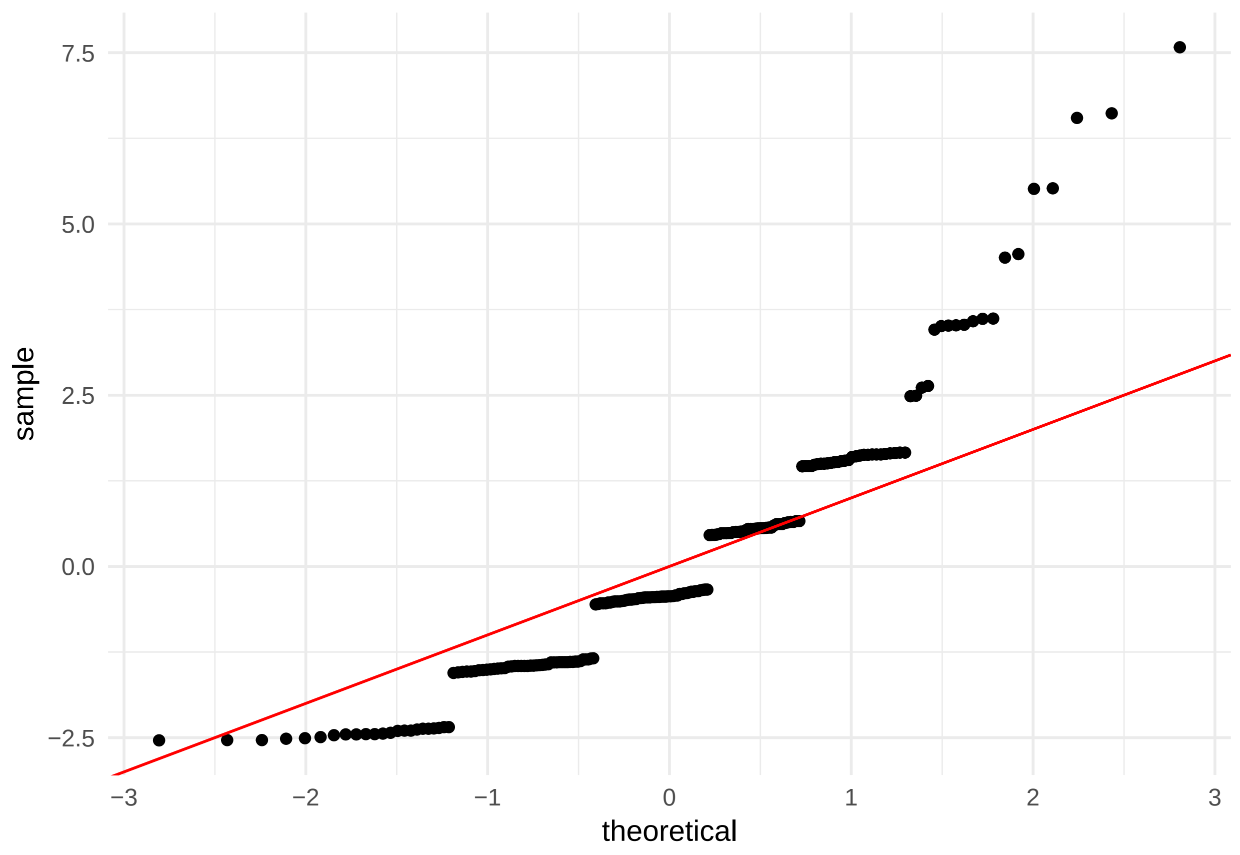
```
D_t %>%
  ggplot(aes(sample = resid)) +
  geom_qq(distribution = qnorm) +
  geom_abline(intercept = 0, slope = 1, col = "red")
```



```r
detach(Chapter_LM)
```

Once mastered, the qq-plot is the swiss knife of Normality check. Next to the subtleties We can also easily discover deviations from symmetry. This is how the residuals of the returns to homepage model look like:

```r
  data_frame(resid = residuals(BrowsingAB$M_age_rtrn)) %>%
  ggplot(aes(sample = resid)) +
  geom_qq(distribution = qnorm) +
  geom_abline(intercept = 0, slope = 1, col = "red")
```



To the left, extreme values have a lower probability than predicted by the Normal distribution, but the right tail are much fatter, one again. We also see how residuals are clumped, which is characteristic for discrete (as compared to continuous) outcome measures. This is poor behaviour of the model and, generally, when a model is severely mis-spedified, neither predictions nor estimate, nor certainty statements can be fully trusted. A model that frequently fits in case of count numbers is Poisson regression, which will enter the stage in chapter @ref(poisson_regression).

Constant variance

In @ref(resid_normality) we have assessed one assumption that underlies all linear models, namely Normal distribution of residuals. The second assumption underlying the linear model random (or residual) is that residual variance is constant throughout the whole range. In both, classic and modern notation this grounds in the there being just a single σ_ϵ defined. However large μ_i is, the dispersion of residuals is not supposed to change.

Before we dive into the matter of checking the assumption let's do a brief reality check using common sense:

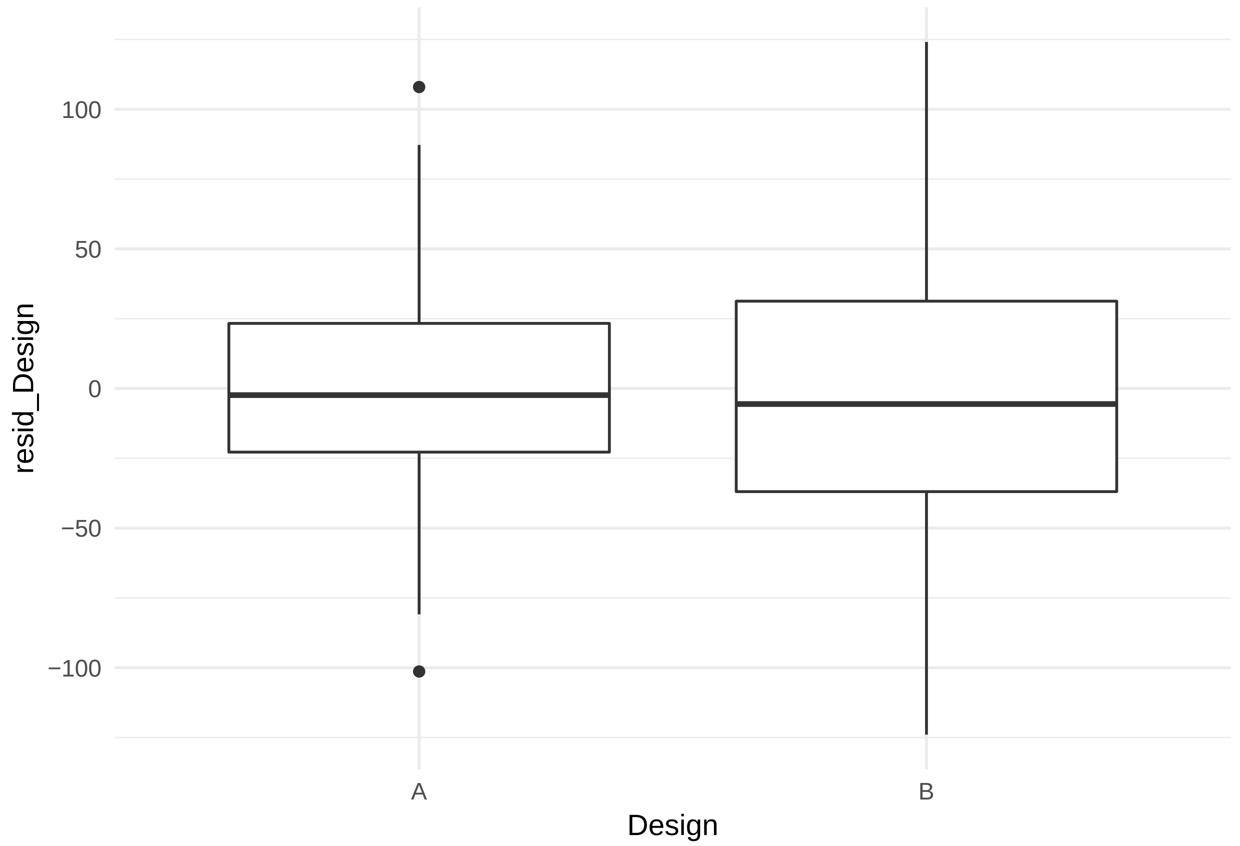
1. Consider people's daily way to work. Suppose you ask a few persons you know: "What is your typical way to work and what is the longest and the shortest duration you remember?". In statistical terms, you are asking for a center estimate and (informal) error dispersion. Is it plausible that a person with typical travel time of 5 minutes experienced the same variation as another person witha typical time of 50 minutes?
2. Consider an experiment to assess a typing training. Is it plausible that the dispersion of typing errors before the training is the same as after the training?

In both cases, we would rather not expect constant variance and it is actually quite difficult to think of a process, where a strong change in average performance is not associated with a change in dispersion. The constant variance assumption, like the normality assumption is a usual suspect when approximating with linear models. We will come back to that down below.

In a similar way, we can ask: can it be taken as granted that residual variance is constant when comparing two or more groups. Would you blindly assume that two rather different designs produce the same amount spread around the average? It may be so, but one can easily think of reasons, why this might be different. We check the situation in the CGM of the BrowsingAB study. Do both design conditions have the same residual variance? Again, we extract the residuals, add them to the data set and produce a boxplot by Design condition:

```
attach(BrowsingAB)
```

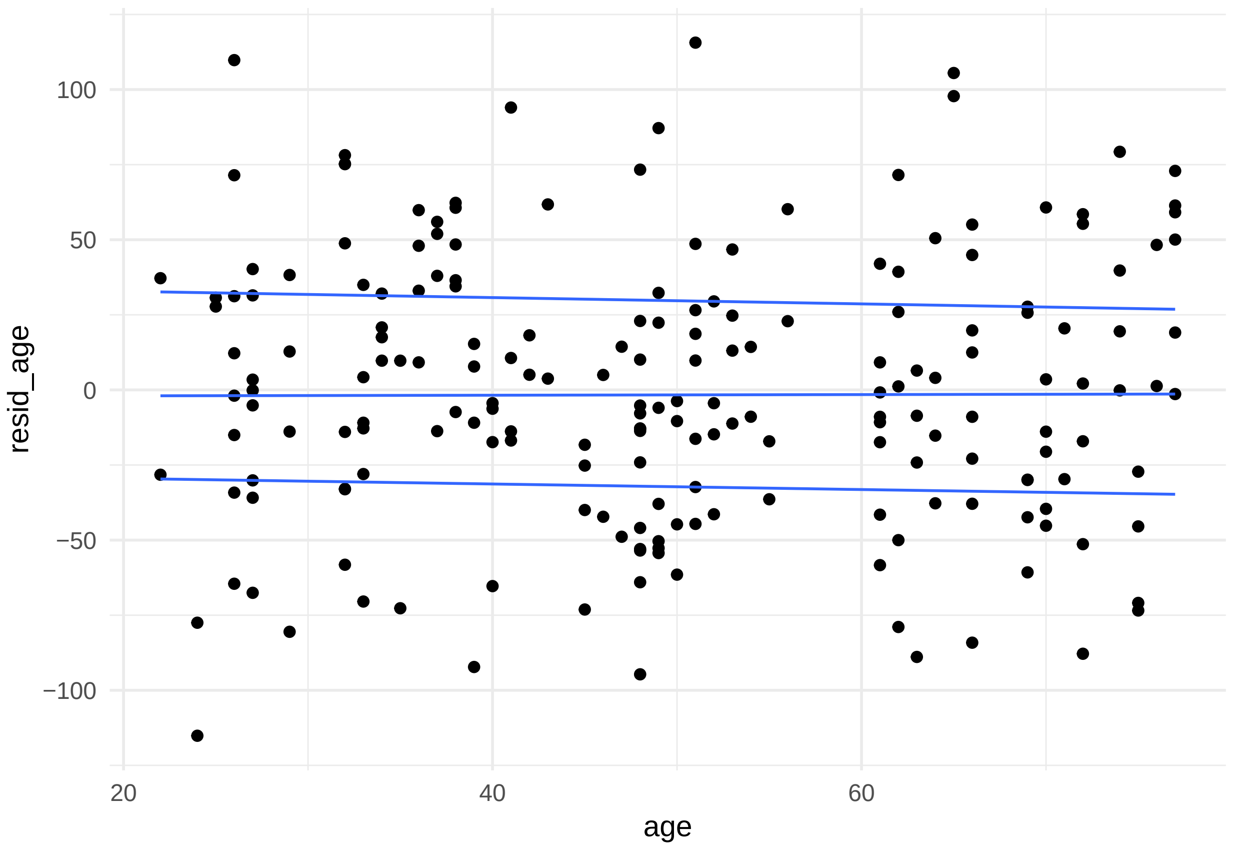
```
BAB1 %>%
  mutate(resid_Design = residuals(M_Design)) %>%
  ggplot(aes(x = Design, y = resid_Design)) +
  geom_boxplot()
```



Both sets of residuals are reasonably symmetric, but it appears that design B produces more widely spread residual. Something in the design causes individual performance to vary stronger from the population mean. The cause of this effect will be disclosed in @ref(differential_design_effects). (In essence, design B is rather efficient to use for younger users, whereas older users seem to have severe issues.)

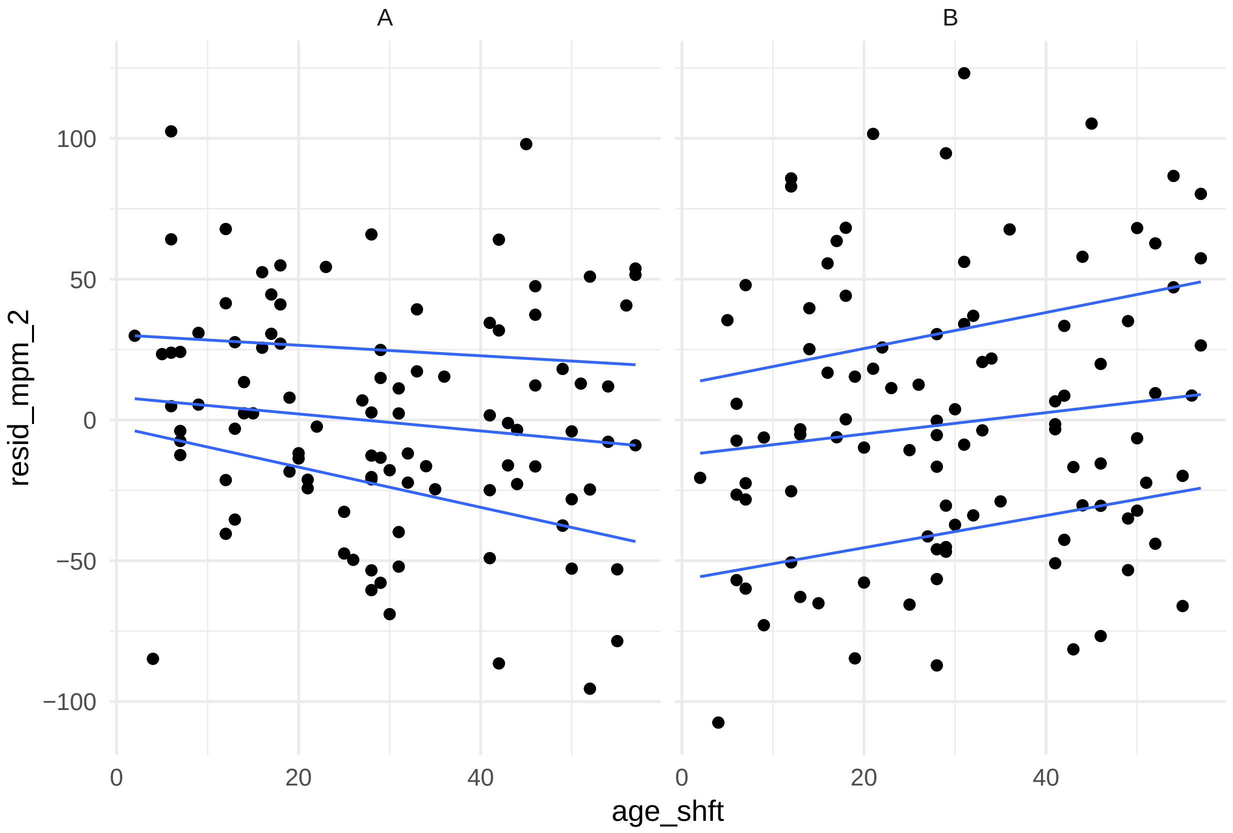
Visual checks of constant variance for factors is straight forward using common boxplots. For continuous predictors, such as age, requires a more uncommon graphical representation known as *quantile plots*. These are not the same as qq plots, but luckily are included with ggplot.

```
BAB1 %>%
  mutate(resid_age = residuals(M_age)) %>%
  ggplot(aes(x = age, y = resid_age)) +
  geom_point() +
  geom_quantile()
```



The quantile plot uses a smoothing algorithm (probably not unlike LOESS) to picture the trend of quantiles (25%, 50% and 75%). Here, the quantiles run almost horizontal and parallel, which confirms constant variance. Taking this as a starting point, we can evaluate more complex models, too. The MPM on age and design, just requires to create a grouped quantile plot. This looks best using facetting, rather than separating by color:

```
BAB1 %>%
  mutate(resid_mpm_2 = residuals(M_mpm_2)) %>%
  ggplot(aes(x = age_shft, y = resid_mpm_2)) +
  facet_grid(~Design) +
  geom_point() +
  geom_quantile()
```



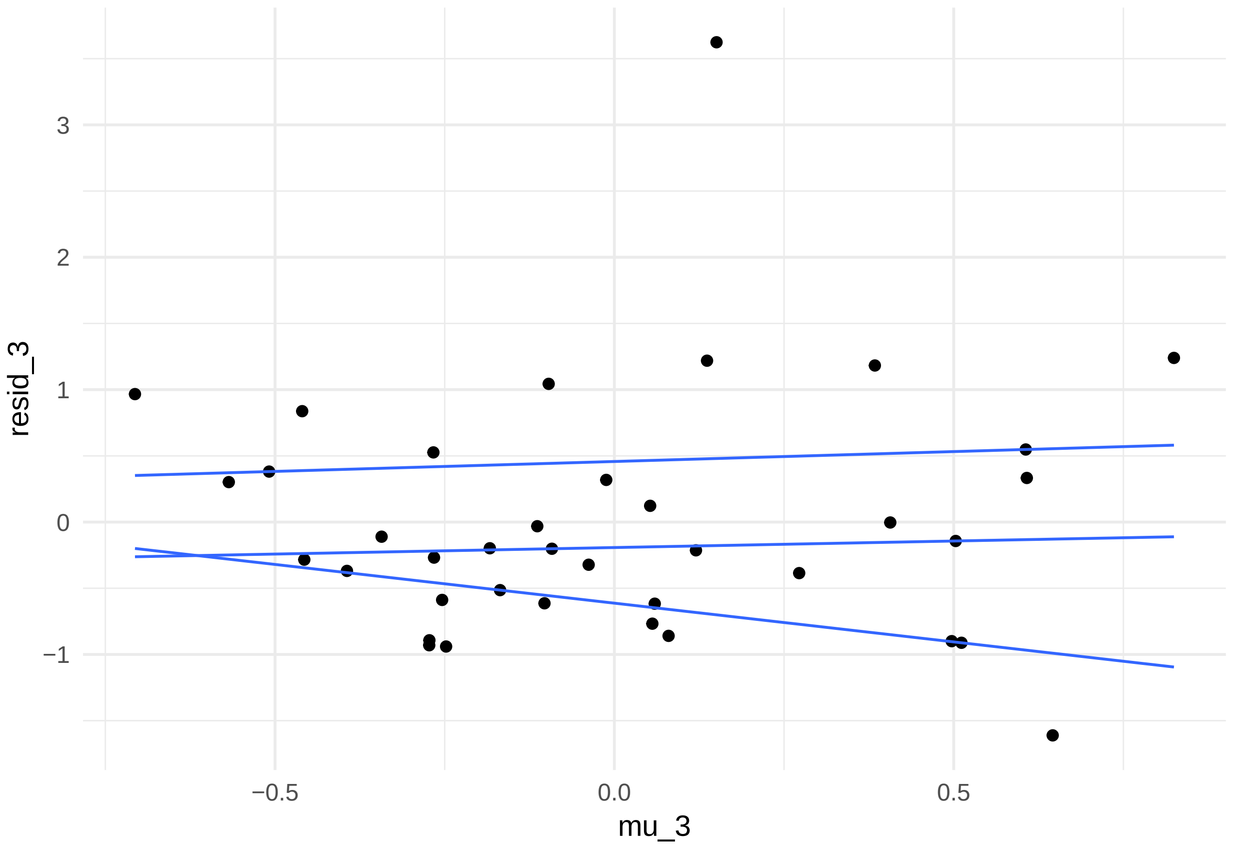
This looks rather worrying. Especially with Design A, the residuals are not constant, but increase with age. In addition, we observe that residuals are not even centered at zero across the whole range. For design A, the residual distribution moves from positive centered to negative centered, design B vice versa. That also casts doubts on the validity of the LRM on age: these contrariwise trends seem to mix into an unsuspicious even distribution. It seems that a lot more has been going on in this study, than would be captured by any of these models.

```
detach(BrowsingAB)
```

Another model type we may want to check with quantile plots is the MRM. With two continuous predictors one might be tempted to think of a 3-dimnensional quantile plot, but this is not recommended. Rather, we can use a generalization of quantile plots, where the x-axis is not mapped to the predictor, directly, but the predicted values μ_i. We assess the residual variance on the MRM model on the AUP study, where resistance to fall for the active user paradox has been predicted by geekism tendencies (gex) and need-for-cognition (ncs):

```
attach(AUP)
```

```
AUP_1 %>%
  mutate(resid_3 = residuals(M_3),
         mu_3 = predict(M_3)$center) %>%
  ggplot(aes(x = mu_3, y = resid_3)) +
  geom_point() +
  geom_quantile()
```



We observe a clear trend in quantiles, with residual dispersion increasing with predicted values. Generally, plotting residuals against predicted values can be done with any model, irrespectively of the number and types of predictors. However, interpretation is more limited than when plotting them against predictors directly. In fact, interpretation boils down to the intuition we introduced at the beginning of the section, that larger outcomes typically have larger dispersion. This is almost always a compelling assumption, or even a matter of underlying physics and, once again, a linear model may or may not be a reasonable approximation. Fortunately, when we turn towards @ref(generalized_linear_models), we will find that these models take provide more reasonable defaults for the relationship between predicted values and dispersion. In contrast, residual variance per predictor allows to discover more surprising issues, such as interaction effects or heterogeneity in groups.

```
detach(AUP)
```

How to plot MPM? [TBC, move to Interaction effects]

Now that we have seen how we can compare on one factor, we can extend the CGM to more factors. With linear models, we can extend the basic CGM to incorporate multiple of such factors. This is called *multifactorial models* (MFM).

Now, we are looking at a slightly more complicated situation, where browsing is predicted by design and education level of participants at the same time.

First, with the *ggplot* system, we plot the new situation. Recall that

1. mapping variables in the data set to elemental properties in the plot, such as: x position, y position, color, shape etc.

2. selecting an appropriate geometry that carries such properties, e.g. points, bars, box plots

Now we can do the two-factorial ANOVA in much the same way as before. We only have to add the predictor *Education* to the model formula. Then we run the MCMC estimation and get the estimates.

[Interaction plots with groups, IA plots with standard debs, mixed IA plots, using the AMM]

Empirical versus statistical control

Fundamental researchers have a knack for the experimental method. An *experiment*, strictly, is a study where you measure the effects of variables you *manipulate*. Manipulation is, almost literally, that it is *in your hands*, who receives the treatment. The fantastic thing about manipulation is that it allows for *causal conclusions*. A *strictly controlled experiment* is when all influencing variables are either manipulated or kept constant. That is an ideal and would not even be the case if you test the same person over-and-over again (like researchers in psychophysics often do). You never jump into the same river twice.

Sometimes an influencing variables lends itself to be kept constant. For example in cognitive psychology experiments, environment and equipment is usually kept constant. For applied research, keeping things constant comes at a major disadvantage: it limits the possible conclusions drawn from the study. Imagine, you tested a smartphone app with participants, all students, comfortably sitting in a quiet environment. Would you dare to make conclusions on how any users perform in real life situations, say while driving a car? When keeping things constant, *ecological validity* and *generalizability* suffer.

In most applied design studies we need ecological validity and generalizability. If performance differs under certain conditions, you certainly want to know that. The solution is to *let conditions vary and record them* as variables, as good as possible. For example, if you were to compare two voice-controlled intelligent agent apps, you could manipulate the ambient noise level, if you are in the lab.

In practically all applied studies, there exist variables, which you cannot manipulate for one of the following reasons. Especially, user traits are impossible to manipulate. If someone has an extrovert character or did a lot of gaming in the past, you cannot change that. Diversity of users is a fact and people come as they are.

Field studies usually aim for high ecological validity. Participants are supposed to use the system in the situations they encounter. If a smartphone app is being used sitting walking, driving or on a secret place, it is crucial to observe all situations. Consider a car navigation system that is tested in a long, lonely highway situation. How much would the results tell your for performance in dense city traffic? Design researchers frequently need results that are highly representative for users and situations of use.

Fundamental lab researchers are afraid of individual differences, too. The reasons are different, though: all non-manipulated influencing factors add noise to the study, which makes it harder to find the effects of interest. While lab researchers do there best to keep the environment constant, they cannot keep all participant traits constant. Lab researchers have two solutions to the problem: matching and randomized control.

With *pair matching*, potentially relevant participant traits are recorded upfront; then participants are assigned to conditions such that groups have about the same composition. For example, one makes sure that the age distribution is about the same and both genders are equally represented. When all other influencing variables are constant between groups, the lab researcher can be sure that the effect is unambiguously caused by the manipulation. So they say and routinely record participants age, gender and nationality.

However, there are better alternatives: the best pair match is the person herself. Experimental studies that expose the same person to several conditions are called *within-subject*. In the special case that all participants encounter all conditions, the variable is *complete within-subject*. [A special case of within-subject design is *repeated measures*.] In the following chapter, we use mixed-effects models [LMM] to deal with within-subject designs, gracefully.

In design research, pair matching applies for situations, where designs are compared. In the simpler situation that a design is evaluated against set standard (e.g. 111 seconds to rent a car), it is more important to do *population matching*. The sample of participants is drawn to be *representative for the target population*. Representativeness comes in two levels: *coverage representation* is reached when all influencing properties have occurred a few times during observation. So, if your target population contains several subgroups, such as age groups, experience or people with different goals, they should all be covered to some extent. *Proportional representation* means all user and situational properties are covered *and* they have about the same proportion in teh sample as in the population.

You can only match what you can measure and you only measure what you expect. Human behavior in everyday life is influenced by many factors in complex ways. Although a plethora of personality inventories exists, doing them all prior to the real study is impossible. It would probably not even be effective. Never have i seen a design research study, where even the most established personality tests explain more than a few percent of variation. As another example, take the primacy effect: what you experienced first, has the strongest influence. In real life, impressions are constantly pouring on people and you will never be able to record and match that to a reasonable extent.

When influencing variables cannot be measured for matching or statistical control, the last resort is *randomized control*. This is a misleading term, insofar as what the researcher actually does is to *let go to chance*. Indeed, if the process of drawing participants and assigning them to manipulations is completely left to chance, then *in the long-term*, the sample will be proportional representative and all groups will have the same composition of traits. *Randomization* works well with larger samples. With small sample, it can still easily happen that one ends up with more or less biased samples or heterogeneous groups. Just by chance, more higher-educated people could have ended up in condition A of BrowsingAB.

Using manipulation, matching or randomization in in-the-wild research may work in some cases. In other cases it will be ineffective or impractical. The ultimate problem is the attempt to keep things constant. In applied design research the questions rarely come down to a "Is A better than B?". If there is an age effect, you may certainly want to know it and see how the design effect compares to it. But, you can only examine what is varied and recorded. The approach of *statistical control* is to record (instead of manipulate) all variables that may influence the results and add them to the statistical model. As we will see now, the linear model puts no limits on the number of predictors. If you believe that user age may play a role for performance, just record it and add it to the model. Statistical control requires at least coverage representation of the sample.

Exercises

1. Rerun `M_3`, this time using the unstandardized resistance scores. Interpret the intercept.
2. The innovators behind the WWW have a long tradition in figurative use of nautical metaphors, like navigator, cybernaut, ... Do people in a world-wide hypermedia system behave like animals roaming known and lesser known territory? Then we would expect performance to be dependent on abilities for spatial cognition, like the visual-spatial working memory. Another theory could draw on the fact that for most websites the written word prevails. Would we not expect people with better verbal processing capabilities to excel? Consider a study `MMN` (for Miller's magic number, an iconic term in working memory research) that examines the influence of working memory capacities on people's browsing performance. Two tests for WM capacity were used: the Corsi task for visual-spatial WM capacity and the Ospan task for verbal WM capacity. Then participants were given five different search tasks on five websites. As a measure of efficiency, total time on task and number of clicks were recorded. As these variables tend to have skewed residual distributions, logarithmic transformation was applied before analysis. *Do LRM on both WM predictors separately, then move on to an MRM.*.

Interaction effects

With the framework of MPM, we can use an arbitrary number of predictors. These can represent properties on different levels, for example, two designs proposals for a website can differ in font size, or participants differ in age. So, with MPM we gain much greater flexibility in handling data from applied design research, which allows us to examine user-design interactions (literally) more closely.

The catch is that if you would ask an arbitrary design researcher:

> Do you think that all users are equal? Or, could it be that one design is better for some users, but inferior for others?

you would in most cases get the answer:

> Of course users differ in many ways and it is crucial to know your target group.

Some will also refer to the concept of usability by the ISO 9241-11, which contains the famous four words:

> "… for a specified user …"

The definition explicitly requires you to state for *for whom* you intended to design. It thereby implicitly acknowledges that usability of a design could be very different for another user group. In other words, statements on usability are by the ISO 9241-11 definition *conditional* on the target user group.

In statistical terms, conditional statements of the form:

> the effect of design changes with (some) user properties

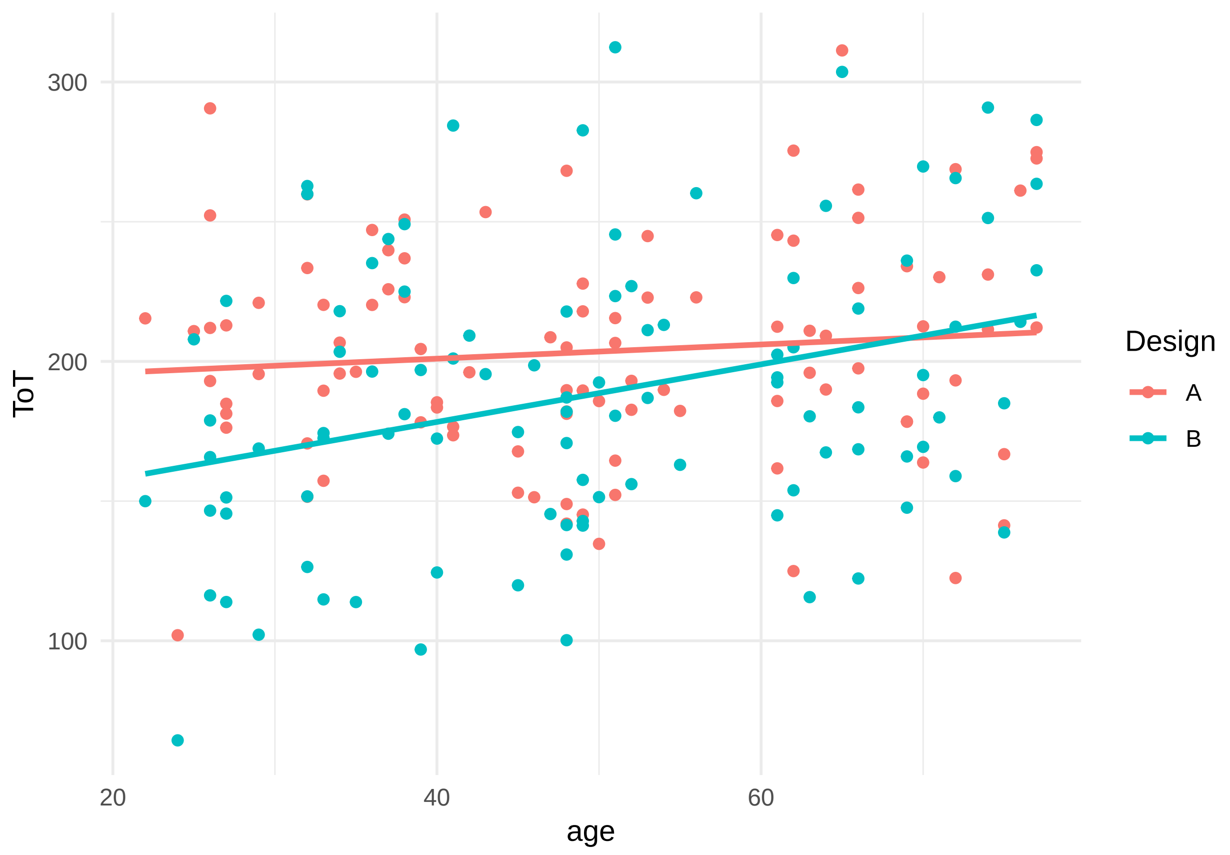
In regression models, conditional statements like these are represented by *interaction effects*. Interactions between user properties and designs are the most genuine in design research, and deserve a neologism: *differential design effects (DDM)*. They come with some of their siblings. *Saturation* occurs when physical (or other) boundaries are reached and the result is less than the sum. *Amplification*, a rare one, is like compound glue: it will harden only if the two components are present. The final section is a plea for interaction effects in theorizing.

Users differ: differential design effects

Do people differ? Yes. If you showed two web designs to a group of stakeholders, asking to choose individually. Is there any chance they would all agree? No. Is the design of universal systems easy? No. The ideal of universal design is to never put a user group at a disadvantage. If it is an ideal, than designs are likely to differ in by how much they accomplish it.

As to age: it is commonly held that older people tend to have lower performance than younger users. A number of factors are called responsible, such as: slower processing speed, lower working memory capacity, lower motor speed and visual problems. Is, perhaps, one of the two designs less of a burden for elderly users? We plot the observed ToT by age, this time forming groups by design.

```r
attach(BrowsingAB)

G_slope_ia <-
  BAB1 %>%
  ggplot(aes(x = age,
             col = Design,
             y = ToT)) +
  geom_point() +
  geom_smooth(method = "lm", se = F)


G_slope_ia
```



As we have seen, with GLM, it is possible to investigate the effect of design properties and user properties simultaneously. For example, assume that main difference between design A and B in the web browsing example is that A uses larger letters than B. Would that create the same benefit for everybody? It is not unlikely, that larger letters only matter for users that have issues with far farsightedness, which is associated with age. Maybe, there is even an adverse effect for younger users, as larger font size takes up more space on screen and more scrolling is required.

We recall the MPM in BrowsingAB: `M_mpm_2` showed a moderate relationship between age and ToT. The design effect was disappointing (`M_Design`): a classic statistician may have called it "significant", but one can hardly claim practical relevance. By adding the interaction effect `Design:age_shft` to the model, we will now investigate how the age effect differs by design. We call this a *differential interaction effect*, as one of the involved

```
M_ia1 <-
  BAB1 %>%
  stan_glm(ToT ~ Design + age_shft + Design:age_shft,
                data = .)

# T_resid <- mutate(T_resid, M_ia1 = residuals(M_ia1))
```

```
T_ia1 <-
  bind_rows(
    posterior(M_mpm_2),
    posterior(M_ia1)) %>%
  fixef()
```

```
T_ia1
```

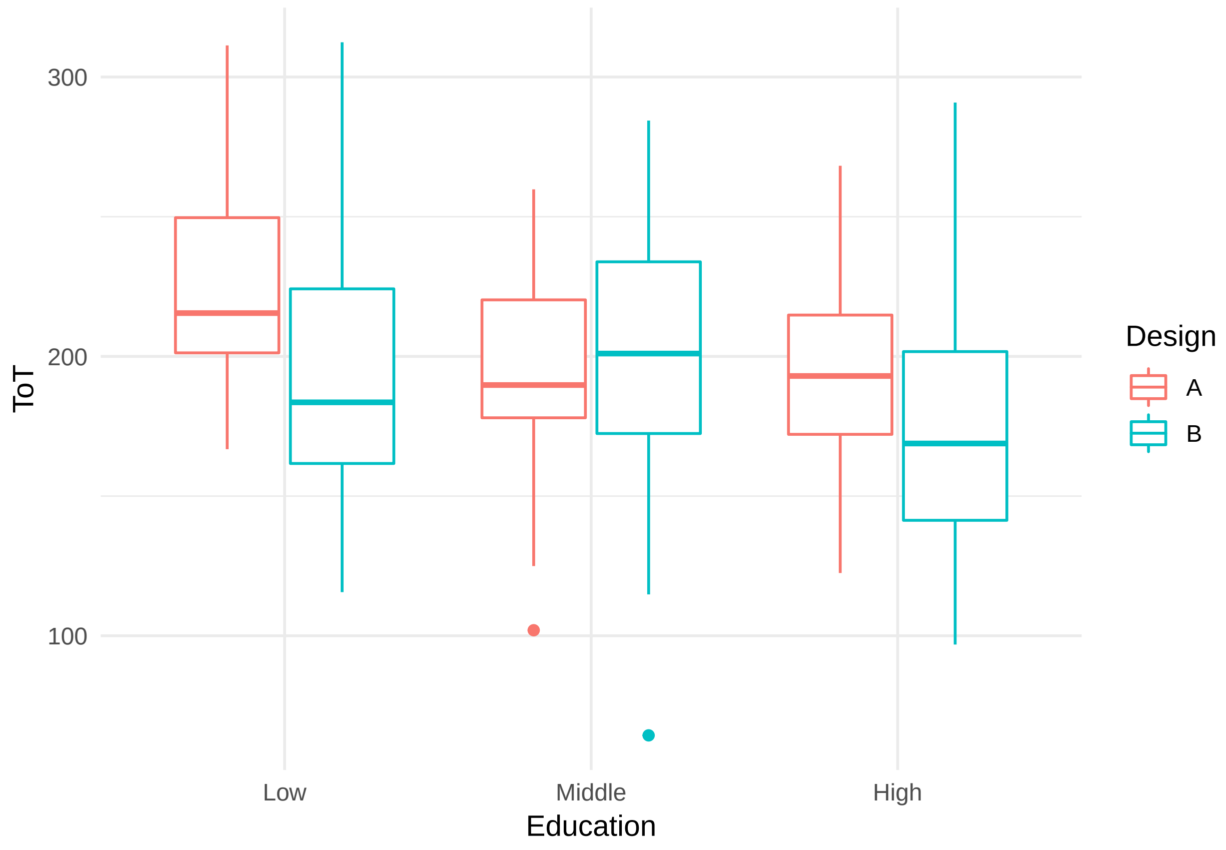
Table 25: Estimates with 95% credibility limits

model	fixef	center	lower	upper
M_ia1	Intercept	195.684	176.708	214.337
M_ia1	DesignB	-37.779	-64.695	-12.813
M_ia1	age_shft	0.259	-0.276	0.798
M_ia1	DesignB:age_shft	0.770	0.056	1.557
M_mpm_2	Intercept	184.208	169.940	198.789
M_mpm_2	DesignB	-14.981	-27.804	-2.140
M_mpm_2	age_shft	0.648	0.254	1.033

The intercept still is the performance of an average twen using reference design A. Compared to the MPM, it is less favorable, with $195.68[176.71, 214.34]_{CI95}$ seconds time-on-task. The effect of design B improved dramatically with the DDM: a twen is now $-37.78[-64.7, -12.81]_{CI95}$ faster with it. Is B the clear winner? It is not, because A has much lower age effect. The age parameter does no longer represent both groups, but just reference A, and it is now very small, $0.26[-0.28, 0.8]_{CI95}$. The final coefficient is *not* the age effect in B, but *the difference* to the reference age effect. With design B, users are getting 1.029 seconds slower per year of life. That is a lot!

If we can do it with covariates, like age, we can do it with factors, too. For example, does the overall improvement from design A to B depend on education level?

```
BAB1 %>%
  ggplot(aes(y = ToT, x = Education, color = Design)) +
  geom_boxplot()
```



Again, we compare the main-effect model to the one with interaction effects.

```
M_mpm_3 <-
  BAB1 %>%
  stan_glm(ToT ~ Design + Education,
                data = .)

M_ia2 <-
  BAB1 %>%
  stan_glm(ToT ~ Design + Education + Design:Education,
                data = .)

# T_resid <- mutate(T_resid, M_ia2 = residuals(M_ia2))
```

```
T_ia2 <-
  bind_rows(
    posterior(M_mpm_3),
    posterior(M_ia2)) %>%
  fixef()


T_ia2
```

Table 26: Estimates with 95% credibility limits

model	fixef	center	lower	upper
M_ia2	Intercept	224.57	208.53	239.20
M_ia2	DesignB	-26.34	-48.10	-4.42
M_ia2	EducationMiddle	-29.51	-51.65	-7.64
M_ia2	EducationHigh	-31.49	-51.70	-10.77
M_ia2	DesignB:EducationMiddle	32.98	1.61	64.12
M_ia2	DesignB:EducationHigh	4.36	-24.50	32.74
M_mpm_3	Intercept	218.83	206.34	231.21
M_mpm_3	DesignB	-15.11	-26.84	-3.27
M_mpm_3	EducationMiddle	-13.46	-28.80	2.44
M_mpm_3	EducationHigh	-29.39	-43.19	-14.46

The model has factors only, so there is a reference group. The intercepts in both models represent low education encountering design A. The main effect designB on low-educated users is present in both models. With the interaction term, design B looks much favorable, $-26.34[-48.11, -4.42]_{CI95}$. The effect of middle education has doubled, whereas it remains stable for high education.

That may appear strange at first, but keep in mind, that by the interaction term, the education main effects are no longer "main": they only refer to group A, now. In the main effects model, the same education effects are assumed for both designs. Here, it is conditional on design, which brings us to the two interaction effects `B:Middle` and `B:High`. These are, once again, differences. The effect of middle education in B is 32.98more seconds than in A (-29.51). There practically is no net effect of middle education in B. In contrast, the high education interaction effect is small, with practically makes `DesignB` a *main effect: it is the same in both groups.*

Count the number of parameters in both models! We have six with interaction and four without. Six is just the number of groups and, indeed, with interaction effects all group means can vary freely. That is best demonstrated by estimating a variant of the model, where six parameters represent six group means, directly. In other words, it is an AGM, without an intercept and without main effects:

```
M_ia3 <-
  BAB1 %>%
  stan_glm(ToT ~ 0 + Design : Education,
               data = .)
# T_resid <- mutate(T_resid, M_ia2 = residuals(M_ia2))
```

We extract the coefficients and, with a little preparation, we create an *interaction plot.* It shows the same pattern as the exploratory plot, but is fully inferential, with error bars indicate the 95% credibility interval.

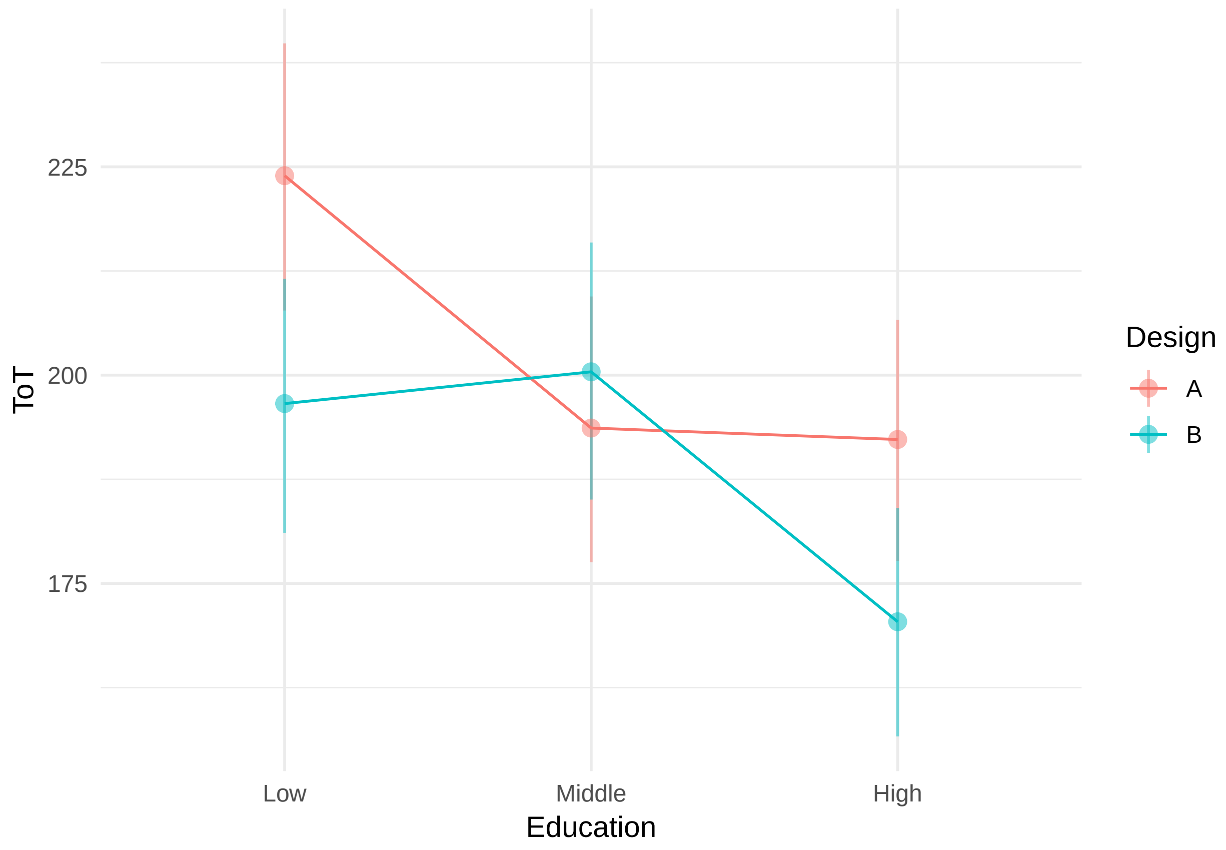
```
P_ia3 <- posterior(M_ia3)
T_ia3 <- fixef(M_ia3)
T_ia3
```

Table 27: Estimates with 95% credibility limits

fixef	center	lower	upper
DesignA:EducationLow	224	208	240
DesignB:EducationLow	197	181	212
DesignA:EducationMiddle	194	178	209
DesignB:EducationMiddle	200	185	216

fixef	center	lower	upper
DesignA:EducationHigh	192	178	207
DesignB:EducationHigh	170	157	184

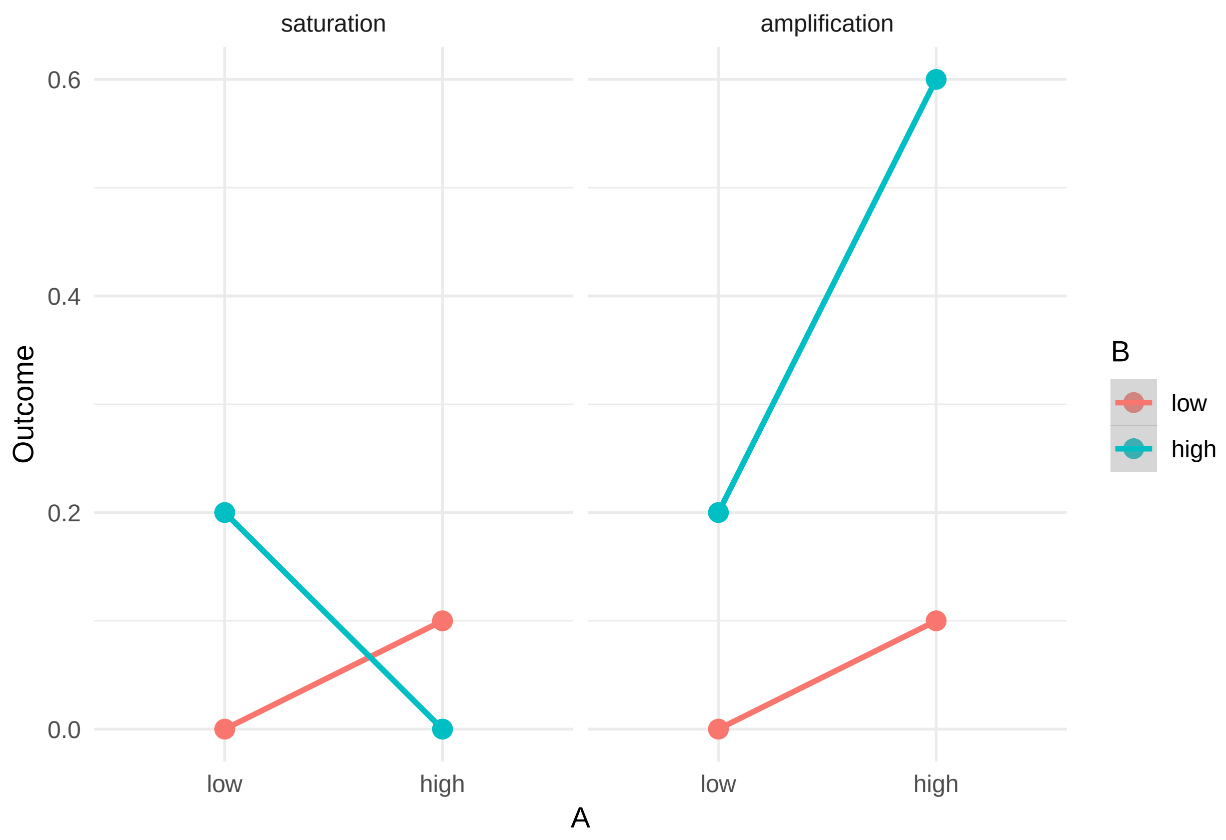
```
T_ia3 %>%
  select(fixef, ToT = center, lower, upper) %>%
  separate(fixef, c("Design", "Education"), ":") %>%
  mutate(Design = str_replace(Design, "Design", ""),
         Education = str_replace(Education, "Education", ""),
         Education = forcats::fct_inorder(Education)) %>%
  ggplot(aes(y = ToT, ymin  = lower, ymax = upper,
             x = Education,
             color = Design, group = Design)) +
  geom_pointrange(alpha = .5) +
  geom_line()
```



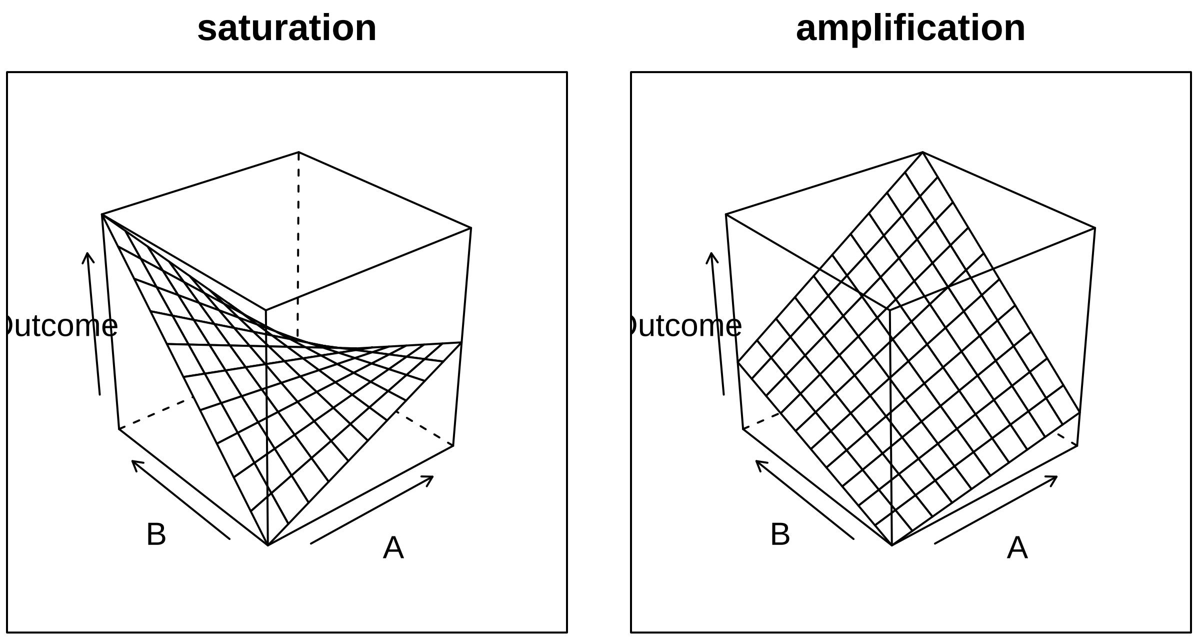
```
detach(BrowsingAB)
```

We can distinguish between *saturation effects* and *amplification effects*.

```
    expand.grid(effect = c("saturation", "amplification"),
                         A = c(0,1),
                         B = c(0,1)) %>%
    join(data.frame(effect = c("saturation","amplification"),
                                beta_1 = c(.1,.1),
                                beta_2 = c(.2,.2),
                                beta_3 = c(-0.3, 0.3)))  %>%
    mutate(Outcome = A * beta_1 + B * beta_2 + A * B * beta_3) %>%
    mutate(B = factor(B, labels = c("low", "high")),
             A = factor(A, labels = c("low", "high"))) %>%
    ggplot(aes(x = A, col = B, y = Outcome)) +
    geom_point(size = 3) +
    geom_smooth(aes(group = B, col= B), method = "lm") +
    facet_grid(.~effect)
```



```
Interactions <- expand.grid(effect = c("saturation", "amplification"),
                        A = seq(0,1, length.out = 11),
                        B = seq(0,1, length.out = 11)) %>%
    join(data.frame(effect = c("saturation","amplification"),
                                beta_1 = c(.1,.1),
                                beta_2 = c(.2,.2),
                                beta_3 = c(-0.3, 0.3)))  %>%
    mutate(Outcome = A * beta_1 + B * beta_2 + A * B * beta_3)

library(lattice)
grid.arrange(
    wireframe(Outcome ~ A + B,
                        data = filter(Interactions, effect == "saturation"),
                          main = "saturation"),
    wireframe(Outcome ~ A + B,
                          data = filter(Interactions, effect == "amplification"),
                          main = "amplification"),
    ncol = 2
)
```



Hitting the boundaries of saturation

Most statistically trained researchers are aware of some common assumptions of linear regression, such as the normally distributed residuals and variance homogeneity. Less commonly regarded is the assumption of linearity, which arises from the basic regression formula:

$$y_i = \beta_0 + \beta_1 x_{1i} ...$$

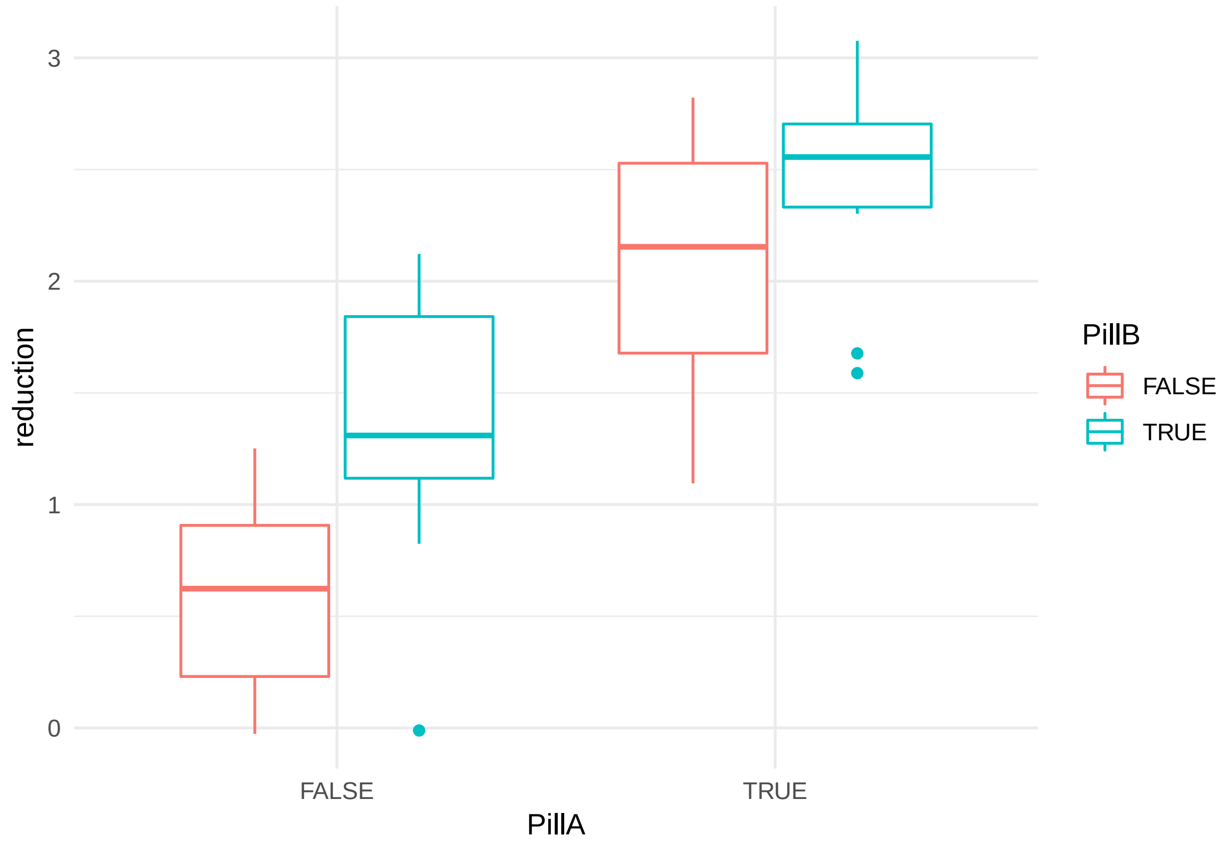
The formula basically says, that if we increase x_1 (or any other influencing variable) by one unit, y will increase by β_1. It also says that y is composed as a mere sum. In this section, we wil discover that these innocent assumption do not hold.

A major flaw with the linear model is that it presumes the regression line to increase and fall infinitely. However, *in an endless universe everything has boundaries.* The time someone needs to read a text is limited by fundamental cognitive processing speed. We may be able to reduce the inconvenience of deciphering small text, but once an optimum is reached, there is no further improvement. Boundaries of performance measures inevitably lead to non-linear relationships between predictors and outcome. Modern statistics knows several means to deal with non-linearity, some of them are introduced in later chapters of this book ([GLM, NLM]).

Still, most researchers use linear models, and it often can be regarded a reasonable approximation, as long as one keeps reasonable distance to the upper and lower boundaries of the outcome.

Before we turn to a genuine design research case, let me explain saturation effects by an example that I hope is intuitive. It boils down to the question: do two headache pills have the double efect than one? Consider a pharmaceutical study on the effectiveness of two pain killer pills A and B. It takes place in the aftermath of a great party on a university campus. Random strolling students are asked to participate. First, they rate their experienced headache on a Likert scale ranging from "fresh like the kiss of morning dew" to "dead highway opossum". Participants are randomly assigned to four groups, each group getting a different combination of pills: A-B, A-Placebo, B-Placebo, Placebo-Placebo. After 30 minutes, experienced headache is measured again and the difference between both measures is taken as the outcome measure, headache reduction. We inspect the position of four group means graphically:

```r
attach(Headache)

Pills %>%
  ggplot(aes(x = PillA, col = PillB, reduction)) +
    geom_boxplot()
```



When neither pill is given a slight spontaneous reduction seems to occur. Both pills alone seem to be effective in their own way, and giving them both has the most beneficial effect. However, it seems that the net effect of B is slightly weaker when administered together with A. Again, when the effect of one predictor depends on the level of another, the effect is called *conditional*.

Now, we will estimate and compare a set of three models, starting with the standard factorial MPM and proceeding to two models with interaction effects. The third one is an AGM with two factors, actually, although it may not seem so at first.

```
M_1 <- stan_glm(reduction ~ 1 + PillA + PillB, data = Pills)
M_2 <- stan_glm(reduction ~ 1 + PillA + PillB + PillA:PillB, data = Pills)
M_3 <- stan_glm(reduction ~ 0 + PillA:PillB, data = Pills)

P <- bind_rows(
  posterior(M_1),
  posterior(M_2),
  posterior(M_3)
)
```

Imagine, the researcher started with the two-way MGM, estimating the fixed effects PillA and PillsB. The result is:

```
T_fixef_1 <- fixef(M_1)
T_fixef_1
```

Table 28: Estimates with 95% credibility limits

fixef	center	lower	upper
Intercept	0.678	0.384	0.975
PillATRUE	1.293	0.959	1.626
PillBTRUE	0.576	0.232	0.907

Given these estimates, we conclude that pill A has an almost double reduction of headache compared to B. The intercept indicates that headache spontaneously diminishes at a rate of $0.68[0.38, 0.98]_{CI95}$. Setting the spontaneous recovery aside, what would you predict the effect to be in the group with both pills?

Thinking linearly you probably came up with:

$$1.87$$

Does that makes sense? Do the effects of headache pills simply add up like this? Consider a scenario, where five headache pills were compared in the same manner. If we assume linear addition of effects, some participants in the group with all pills could even experience the breathtaking sensation of *negative* headache. Certainly, the effect is not linear. But, is it in the long run? In some Hollywood movies the wounded hero consumes handful of pills before the showdown (With the exception of Leon, the Profi, where it is the scary Gary Oldman consuming some harder stuff). With two pills, how close to the boundaries are we? We can just test that by introducing an *interaction term* to the model: `PillaA:PillB`.

```
T_fixef_2 <-  fixef(M_2)
T_fixef_2
```

Table 29: Estimates with 95% credibility limits

fixef	center	lower	upper
Intercept	0.592	0.252	0.940
PillATRUE	1.469	0.976	1.940
PillBTRUE	0.746	0.270	1.226
PillATRUE:PillBTRUE	-0.359	-1.017	0.315

We first examine the main effects for both pills. Interestingly, both effects are now considerably stronger than before. Taking either pill alone is more effective than we thought from the multiple CGM. The fourth coefficient (`PillATRUE:PILLBTRUE`) is the interaction term. With the updated model we, again, predict the net headache reduction when both pills are given. This is as simple as summing up all coefficients from intercept to interaction. The result is a combined effect of both main effects, *reduced* by 0.36. As we would expect, the effectiveness of two pills is not their sum, but less. One can have headache to a certain degree or no headache at all. If it's gone, any more pills have no additional effects. This is called an *saturation effect*: the more is given, the closer it gets to teh natural boundaries and the less it adds. Let me emphasize that this is not some strange (and therefore interesting) metabolic interaction between pharmaceuticals. Saturation effects must not be confused with any "real" interaction, that is theoretically interesting. Still, as we have seen, ignoring saturation effects results in severely biased estimates.

Back to design research! Imagine, you are testing the influence of font size on reading performance. In your experiment, you found that increasing from 8pt to 12pt cuts the required reading time by 10 seconds. According to the linear model, you had to expect the reading time to decrease by another 10 seconds, when enlarging to 16pt.

```
detach(Headache)
```

```
attach(Reading)

D_reading_time <-
  data_frame(font_size = c(4, 8, 12, 16, 24, 28),
             observed_time = c(NA, 40, 30, NA, NA, NA),
             predicted_time = 60 - font_size/4 * 10) %>%
  as_tbl_obs()

D_reading_time
```

Table 30: Data set: showing 6 of 6 observations

Obs	font_size	observed_time	predicted_time
1	4	NA	50
2	8	40	40
3	12	30	30
4	16	NA	20
5	24	NA	0
6	28	NA	-10

It should be clear by now, that these expectations have no ground: for normally sighted persons, a font size of 12 is easy enough to decipher and another increase will not have the same effect. Taking this further, one would even arrive at absurdly short or impossible negative reading times. At the opposite, a font size of four point may just render unreadable on a computer screen. Instead of a moderate increase by 10 seconds, participants may have to decipher and guess the individual words, which will take much longer.

Researching the effect of font sizes between 8pt and 12pt font size probably keeps the right distance, with approximate linearity within that range. But what happens if you when you bring a second manipulation into the game, that has a functionally similar effect? A likely outcome is that the fundamental assumption of *predictors sum up* no longer holds, but *saturation* occurs.

Let's turn to a fictional, yet realistic problem in design research. Design of systems is a matter of compromises. A common conflict of interests is between the aesthetic appearance and the ergonomic properties. Consider the typesetting on web pages. Many commercial websites are equally concerned about the ease of comprehending information and positive emotional response. From an ergonomic point of view, the typesetting of longer text simply requires you to choose the best screen-readable font, find an optimal font size and maximize contrast. If you were now to suggest typesetting a website in black Arial on white background, you are in for trouble with your graphic designer or, much worse, the manager responsible for corporate impression management. In any case, someone will insist on a fancy serif font (which renders rather poorly on low resolution devices) in an understating blueish-grey tone. For creating a relaxed reading experience, the only option left is to increase the font size. This option is limited by another compromise, though, as too large letters requires excessive scrolling.

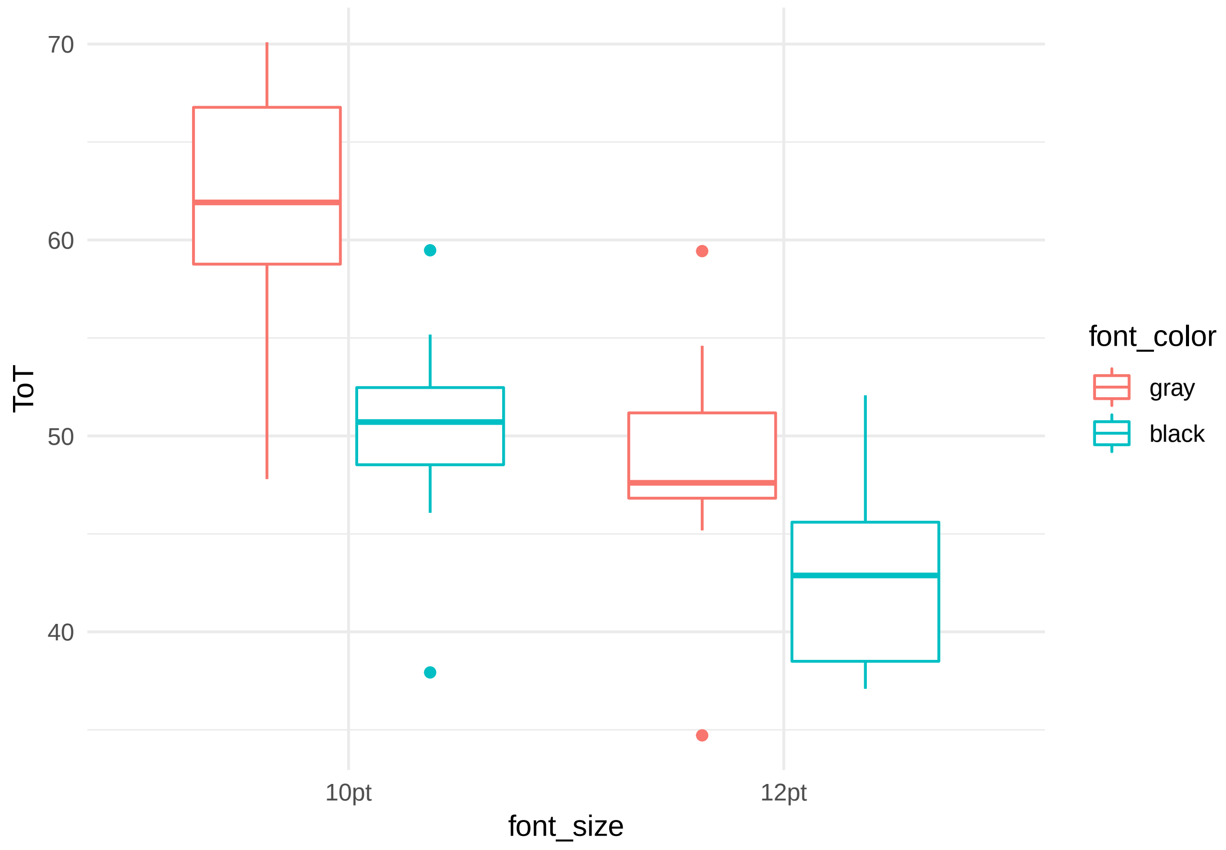
The question arises: can one sufficiently compensate lack of contrast by setting the text in the maximum reasonable font size 12pt, as compared to the more typical 10pt? In the fictional study Reading, this is examined in a 2x2 between-subject design: the same page of text is presented in four versions, with either 10pt or 12pt, and grey versus black font color. Performance is here measured as time-on-task of reading the full page.

```
D_1
```

Table 31: Data set: showing 8 of 40 observations

Part	font_size	font_color	mu	ToT
2	12pt	gray	48	45.2
10	12pt	gray	48	47.7
13	10pt	gray	60	53.1
20	12pt	gray	48	54.6
25	10pt	black	50	59.5
29	10pt	black	50	52.3
33	10pt	black	50	55.2
34	12pt	black	46	43.0

```
D_1 %>%
  ggplot(aes(col = font_color,
             x = font_size,
             y = ToT)) +
  geom_boxplot()
```



We see immediately, that both design choices have a an impact: more contrast, as well as larger letters lead to improved reading performance. But, do they add up? Or do both factors behave like headache pills, where more is more, but less than the sum. Clearly, the 12pt-black group could read fastest on average. So, neither with large font, nor with optimal contrast alone has the design reached maximum performance, i.e. saturation. Still, there could be an interaction effect, that gradually reduces the positive effect of large font in a high-contrast design. We run two regression model, one with both main effects and one that adds an interaction term. We extract the coefficients from both models and view them side-by-side:

```
M_1 <-
  D_1 %>%
  stan_glm(ToT ~ font_size + font_color,
           data = .)

M_2 <-
  D_1 %>%
  stan_glm(ToT ~ font_size + font_color + font_size : font_color,
           data = .)
```

```
T_read_fixef <-
  right_join(select(fixef(M_1), fixef, M_1 = center),
             select(fixef(M_2), fixef, M_2 = center),
             by = "fixef")

T_read_fixef
```

Table 32: Estimates with % credibility limits

fixef	M_1	M_2
Intercept	59.79	61.17
font_size12pt	-9.91	-12.45
font_colorblack	-8.23	-10.78
font_size12pt:font_colorblack	NA	5.22

The estimates confirm, that both manipulations have a considerable effect. And, there is an interaction effect as well, correcting the additive effect of font color and size. The combined effect of high contrast and large font is therefore

$$\mu_{12pt,black} = 61.172 + -12.45 + -10.783 + 5.222 = 43.161$$

In the research scenario, that was not the question, strictly, as we were primarily interested in comparing the effects of font size and contrast. Also, if we see the credibility interval of the interaction effect it is not highly certain ($5.22[-2.25, 12.57]_{CI95}$)) Still, including the interaction term is the right choice, for two reasons: first, both manipulations influence the same cognitive processing, as they both improve visual clarity, with the effect of better visual letter recognition. From a theoretical perspective, we would thus expect saturation. This is simply the application of prior knowledge and in a perfect world one would use a prior distribution for the interaction term, creating a more certain estimate. That being said, the data set is synthetic, and the simulation definitely included the interaction effect. Second, in Table [XY], all other coefficients change considerably with introduction of the interaction effect. Especially, the effects of the two manipulations get considerably under-estimated.

Why are the main effects under-estimated, when not including the interaction term? The pure main-effects model has three parameters. This allows it to represent the same number of independent group means. Formally, the number of groups in the study is four. However, the three-parameter model assumes that the fourth group (black, 12pt) can sufficiently be specified by the existing three parameters. If saturation occurs, the group of participants in the 12pt group is not homogeneous: in the grey group, they experience a stronger improvement than in the black group. The three parameter model is forced to make the best out of situation and adjusts the net effect of font size to be slightly lower than it actually is. The same occurs for the font color effect.

Interaction effects are notoriously neglected in research, and they are hard to grasp for audience with or without a classic statistics education. It is therefore highly recommended to illustrate interaction effects using interaction plots. To begin with, an interaction plot for the 2x2 design contains the four estimated group means. These can be computed from the linear model coefficients, but there is an alternative way: we modify the model, such that it represents the four group means, directly. The re-parametrized model has no intercept (if we want the group means directly, we need no reference group), and no main effects. The model contains just the raw interaction effect, which results in an estimation of the four group means:

```r
M_3 <-
  D_1 %>%
  stan_glm(ToT ~ 0 + font_size : font_color,
           data = .)
```

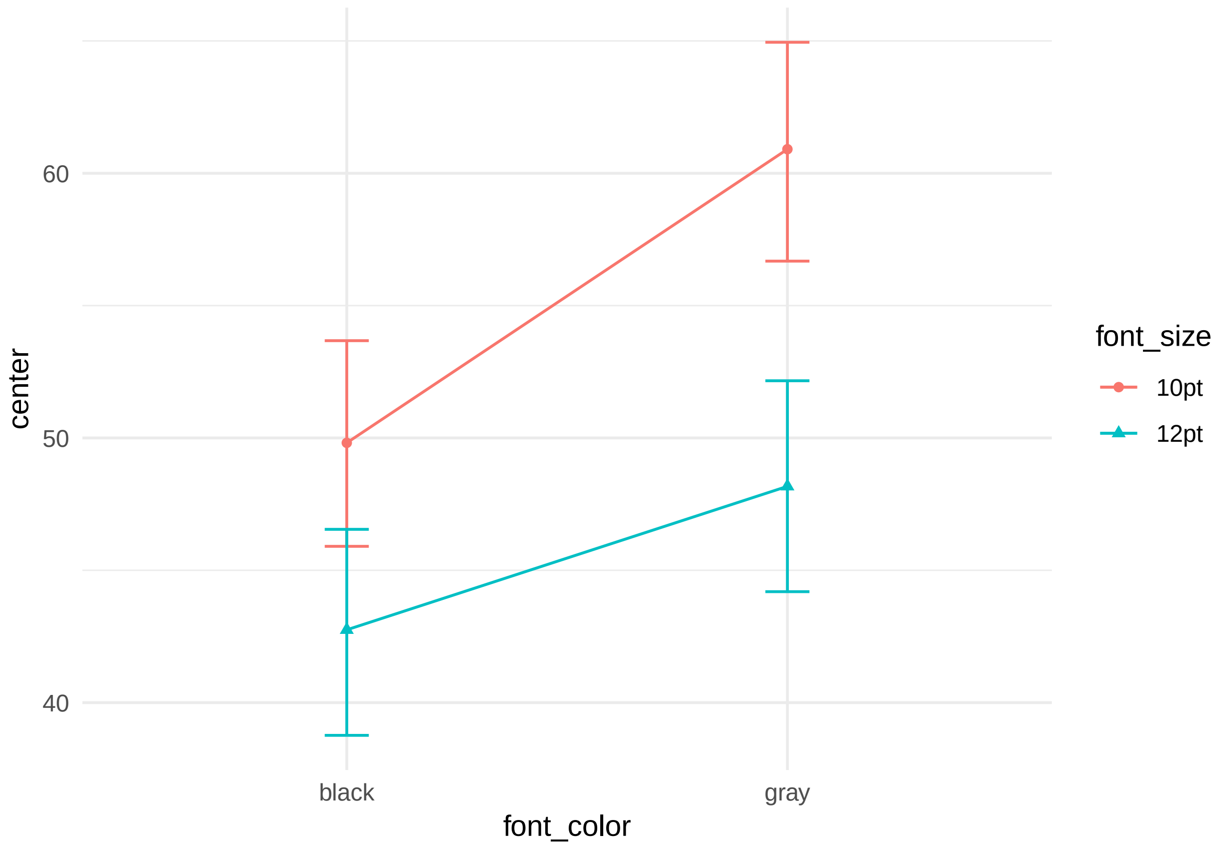
```r
fixef(M_3)
```

Table 33: Estimates with 95% credibility limits

fixef	center	lower	upper
font_size10pt:font_colorgray	60.9	56.7	64.9
font_size12pt:font_colorgray	48.2	44.2	52.2
font_size10pt:font_colorblack	49.8	45.9	53.7
font_size12pt:font_colorblack	42.8	38.8	46.5

With some basic data transformation on the coefficient table, we create a data frame that encodes the two factors separately. Note how `separate` is used to split the coefficient names into group identifiers. Using `mutate` and `str_replace`, the group labels are stripped the factor names. The resulting coefficient table serves as input to ggplot, creating an interaction plot with overlayed credibility intervals, using `geom_errorbar`.

```
T_2x2_mu <-
  fixef(M_3) %>%
  separate(fixef, c("font_size", "font_color"), sep = ":") %>%
  mutate(font_size = str_replace(font_size, "font_size", ""),
         font_color = str_replace(font_color, "font_color", "")) %>%
  as.data.frame()

G_interaction <-
  T_2x2_mu %>%
  ggplot(aes(x = font_color,
             color = font_size, shape = font_size,
             y = center,
             ymin = lower,
             ymax = upper)) +
  geom_point() +
  geom_line(aes(group = font_size)) +
  geom_errorbar(width = .1)

G_interaction
```



```r
detach(Reading)
```

From Figure XY we can easily spot the two main effects, and how they go together. The researcher would conclude that with the desired gray font, increasing font size partly compensates for the lack of contrast. At the same time, we see that some saturation occurs, but still, from a purely ergonomic perspective, large font and high contrast is the preferred design.

We have seen in the Headache example that interaction effects occur as non-linearity. The more a participant approaches the natural boundary of zero headache, the less benefit is created by additional effort. This we call *saturation*. Saturation is likely to occur when multiple factors influence the same cognitive or physical system or functioning. In quantitative comparative design studies, we gain a more detailed picture on the co-impact of design interventions and can come to more sophisticated decisions.

In the Reading case, overall utility is a compound of ergonomic quality and aesthetic appeal. It was assumed that a gray-on-white color scheme is more appealing. The researcher would choose the gray-12pt design: higher contrast would increase ergonomic value, but too much at the expense of appeal. Of course, in a perfect world, one would add emotional appeal as a second measure to the study.

If we don't account for saturation by introducing interaction terms, we are prone to underestimate the net effect of any of these measures, and may falsely conclude that a certain measure is ineffective. Consider a large scale study, that assesses the simultaneous impact of many demographic variables on how willing customers are to take certain energy saving actions in their homes. It is very likely that subsets of variables are associated with similar cognitive processes. For example, certain action requires little effort (such as switching off lights in unoccupied rooms), whereas others are time-consuming (drying the laundry outside). At the same time, customers may vary in the overall eagerness (motivation). For high effort actions the impact of motivation level probably makes more of a difference as when effort is low. Not including the interaction effect would result in the false conclusion that suggesting high effort actions is rather ineffective.

More than the sum: amplification

Saturation effect occur, when multiple impact factors act on the same system and work in the same direction. When reaching the boundaries, the change per unit diminishes. These impact factors can be similar (headache pills) or unsimilar, like font size and contrast. Still, they are to some extent exchangeable. We can compensate lack of contrast with larger font size.

Amplification interaction effects are the opposite: They occur, when two (or more) factors impact different cognitive processes in a processing chain. Conceiving good examples for amplification effects is far more challenging as compared to saturation effects. Probably this is because saturation is a rather trivial phenomenon, whereas amplification involves some non-trivial orchestration of cognitive or physiological subprocesses. Here, a fictional case on technology acceptance will serve to illustrate amplification effects. Imagine a start-up company that seeks funding for a novel augmented reality game, where groups of gamers compete for territory. For their fund raising endeavor they need to know their market potential, i.e. which fraction of the population is potentially interested. The entrepreneurs have two hypotheses they want to verify:

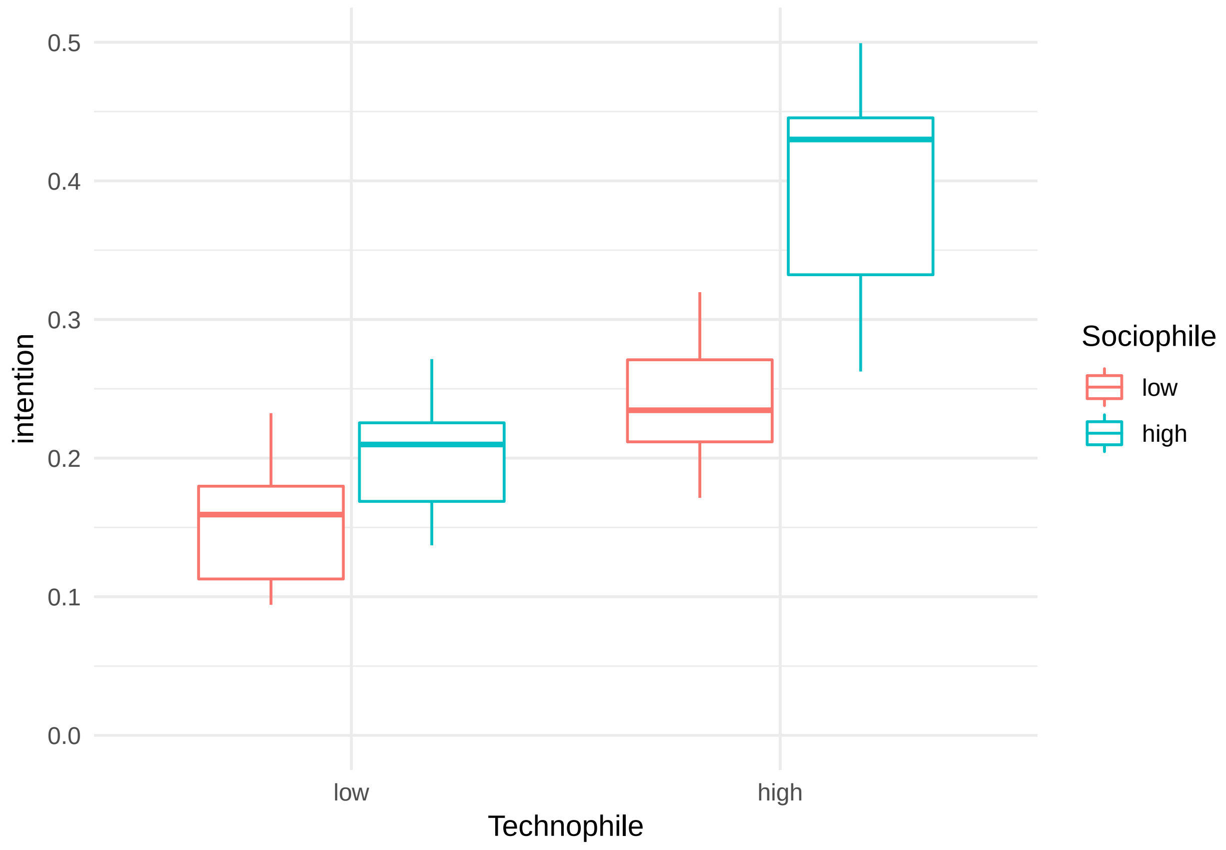
1. Only technophile persons will dare to play the game, because it requires some top-notch equipment.
2. The game is strongly cooperative and therefore more attractive for people with a strong social motif.

In their study they asked a larger set of participants to rate their technophily and sociophily. Then they were given a description of the planned game and were asked how much they intend to participate in the game.

While the example primarily serves to introduce amplification effects, it is also an opportunity to get familiar with interaction effect between metric predictors. While this is not very different to interaction effects on groups, there are a few peculiarities, one being that we cannot straight-forwardly make an exploratory plot. For factors we have used box plots, but these do not apply for metric variables. In fact, it is very difficult to come up with a good graphical representation. One might think of 3D wire-frame plots, but these transfer poorly to the 2D medium of these pages. Another option is to create a scatter-plot with the predictors on axis and encode the outcome variable by shades or size of dots. These options may suffice to see any present main effects, but are too coarse to discover subtle non-linearity. The closest we can get to a good illustration is create groups and continue as usual. Note that turning metric predictors into factors is just a hack to create exploratory graphs. By no means do I intend to corroborate the use of group-mean models on metric data.

```
attach(AR_game)
```

```
D_1 %>%
  mutate(Sociophile = forcats::fct_rev(ifelse(sociophile > median(sociophile), "high", "low")),
         Technophile = forcats::fct_rev(ifelse(technophile > median(technophile), "high", "low"))) %>%
  ggplot(aes(y = intention, x = Technophile, col = Sociophile)) +
  geom_boxplot() +
  ylim(0, 0.5)
```



From the boxplot it seems that both predictors have a positive effect on intention to play. However, it remains vague if there is an interaction effect. In absence of a good visualization, we have to rely fully on the numerical estimates of a linear regression with interaction effect.

```
M_1 <-
  D_1 %>%
  stan_glm(intention ~ sociophile * technophile,
           data = .)
```

```
T_1 <- fixef(M_1)
T_1
```

Table 34: Estimates with 95% credibility limits

fixef	center	lower	upper
Intercept	0.273	0.259	0.288
sociophile	0.113	0.075	0.153
technophile	0.183	0.152	0.212
sociophile:technophile	0.165	0.082	0.246

The regression model confirms that sociophilia and technophilia both have a moderate effect on intention. Both main effects are clearly in the positive range. Yet, when both increase, the intention increases overlinearly. The sociophile-technophile personality is the primary target group.

While plotting the relationship between three metric variables is difficult, there are alternative ways to illustrate the effect. For example, we could ask: how does intention change by .1 unit sociophilia for two imaginary participants with extreme positions on technophilia (e.g., .2 and .8). We could calculate these values arithmetically using the model formula and the center estimates. The better and genuinely Bayesian way of doing it is sampling from the *posterior predictive distribution* (PPD). This distribution is generated during parameter estimation. While the posterior distribution (PD) represents the predictors, the PPD is linked to the outcome variable. More specifically, the PPD is the models best guess of the true value μ, separated from the random component. Once the posterior has been estimated, we can draw from it with any values of interest. As these can be combinations of values that have never been observed, we can truly speak of prediction. Regression engines usually provide easy means to simulate from a PD and generate predictions. In the following example, we simulate some equidistant steps of sociophilia for three participants with technophilia scores from .1 to .9.

```
D_2 <-
  mascutils::expand_grid(technophile = seq(.1, .9, by = .1),
                                   sociophile = seq(.3, .6, by = .1)) %>%
  arrange(technophile)

T_1_pred <-
  post_pred(M_1, newdata = D_2, thin =2) %>%
  predict()

D_2 <-
  D_2 %>%
  mutate(intention = T_1_pred$center)

D_2 %>%
  mutate(technophile = as.factor(technophile)) %>%
  ggplot(aes(x = sociophile, col = technophile, y = intention)) +
  geom_point() +
  geom_line(aes(group = technophile))
```

The effect is not stunning, but visible. The lines diverge, which means they have different slopes. With high technophilia, every (tenth) unit of sociophilia has a stronger effect on intention to play.

```
detach(AR_game)
```

Amplification effects are like two-component glue. When using only one of the components you just get a smear. Putting both togeher gives a strong hold. There also is a parallel in Boolean algebra. Recall, the Boolean OR operator returns `TRUE` when one of the operands is `TRUE`, whereas AND requires both operands to be `TRUE`.

```
T_bool_interaction <-
data_frame(A = c(F, F, T, T),
           B = c(F, T, F, T)) %>%
  as_data_frame() %>%
  mutate("A OR B" = A | B) %>%
  mutate("A AND B" = A & B)

kable(T_bool_interaction)
```

A	B	A OR B	A AND B
FALSE	FALSE	FALSE	FALSE

A	B	A OR B	A AND B
FALSE	TRUE	TRUE	FALSE
TRUE	FALSE	TRUE	FALSE
TRUE	TRUE	TRUE	TRUE

The Boolean OR is an extreme case of saturation. Once one of the factors is present, the result is "positive" and introducing the other one has absolutely no additional effect. Another basic Boolean operation is $AANDB$ with quite the opposite effect: any of the two factors only has an effect, if the other is present, too. Clearly, this is a conditional effect. Rarely will we encounter non-trivial cases where the results are so crisp as in the Boolean world.

Theoretically interesting interaction effects [EDIT]

Explaining or predicting complex behavior with psychological theory is one corner stone of design research. Unfortunately, it is not that easy. While design is definitely multifactorial, with a variety of cognitive processes, individual diferences and behavioural strategies, few psychological theories cover more than three associations between external or individual conditions and behavior. The design researcher is often forced to enter a rather narrow perspective, or knit a patchwork model from multiple theories. Such a model can either be loose, making few assumptions on how the impact factors interact which others. A more tightened model frames multiple impact factors into a conditional network, where impact of one factor can depend on the overall configuration. A classic study will now serve to show how interaction effects and theoretical reasoning go together.

Vigilance is the ability to endure in attention for rarely occurring events. Think of truck drivers on lonely night rides, where most of the time they spend keeping the truck on a straight 80km/h course. Only every now and then is the driver required to react to an event like braking lights flaring up ahead. Vigilance tasks are among the hardest thing to ask from a human operator, yet, they are safety relevant in a number of domains. Keeping up vigilance most people perceive as tiring, and vigilance deteriorates with tiredness.

Several studies have shown that reaction time at simple tasks increases when people are tired. The disturbing effect of noise has been documented as well. A study by Corcoran (1961) examined the simultaneous influence of sleep deprivation and noise on a rather simple reaction task. They ask:

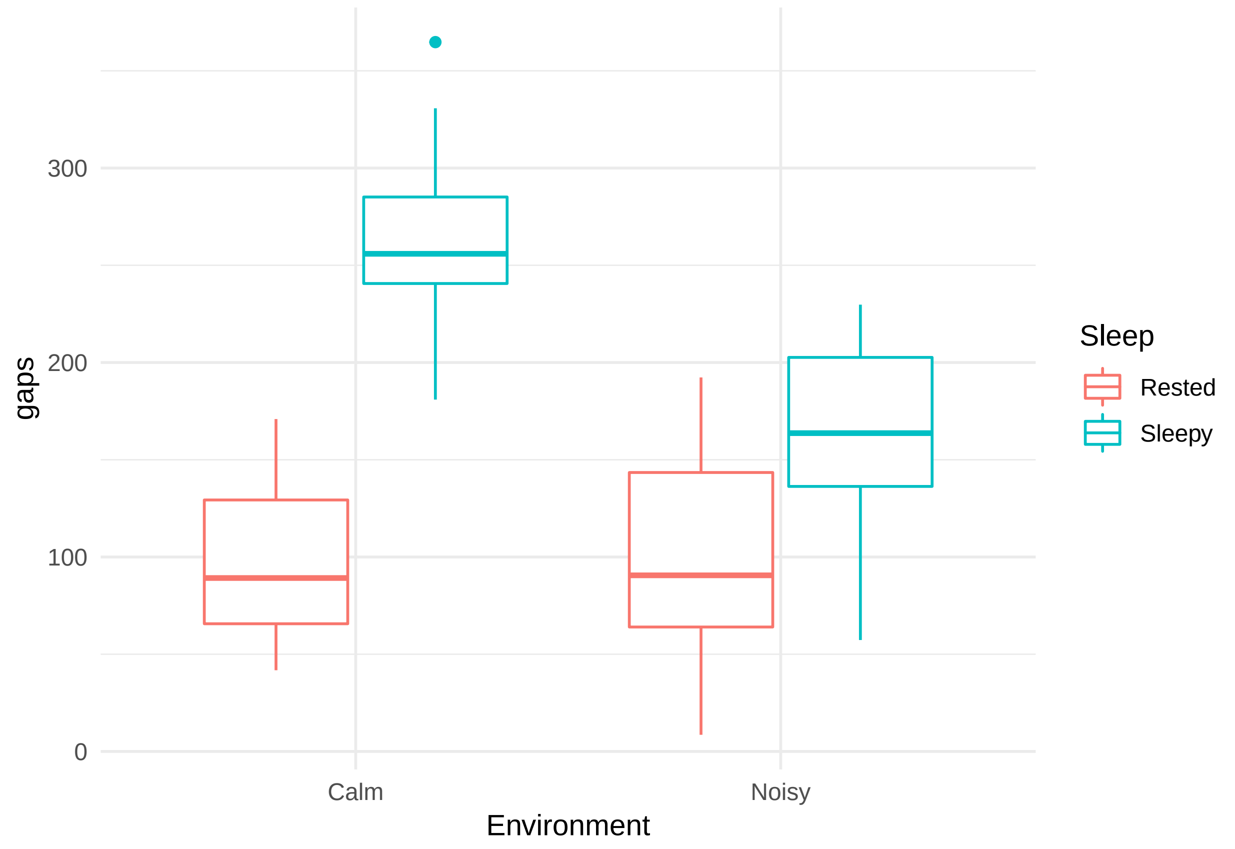
> will the effects of noise summate with those of loss of sleep to induce an even greater performance decrement or will noise subtract from the performance decrement caused by loss of sleep?

The central argument is that sleep deprivation deteriorates a central nervous arousal system. In consequence, sleep deprived persons cannot maintain the necessary level of tension that goes with the task. Noise is a source of irritation and therefore usually reduces performance. At the same time, noise may have an arousing effect, which may compensate for the loss of arousal due to sleep deprivation. To re-iterate on the headache pills analogy, noise could be the antidote for sleepiness.

The Sleep case study is a simplified simulation of Corcoran's results. Participants were divided into 2x2 groups (quiet/noisy, rested/deprived) and has to react to five signal lamps in a succession of trials. In the original study, as performance measure gaps were counted, which is the number of delayed reactions ($> 1500ms$).

```r
attach(Sleep)
```

```r
G_expl <-
  D_1 %>%
  ggplot(aes(x = Environment,
            color = Sleep,
            y = gaps)) +
  geom_boxplot()
G_expl
```



Using a 2x2 model including an interaction effect, we examine the conditional association between noise and sleepiness. The * operator in the model formula is an abbreviation for the fully factorial term `Environment + Sleep + Environment:Sleep`.

```
M_1 <-
  D_1 %>%
  stan_glm(gaps ~ Environment * Sleep, data = .)
```

```
T_fixef <-  fixef(M_1)
T_fixef
```

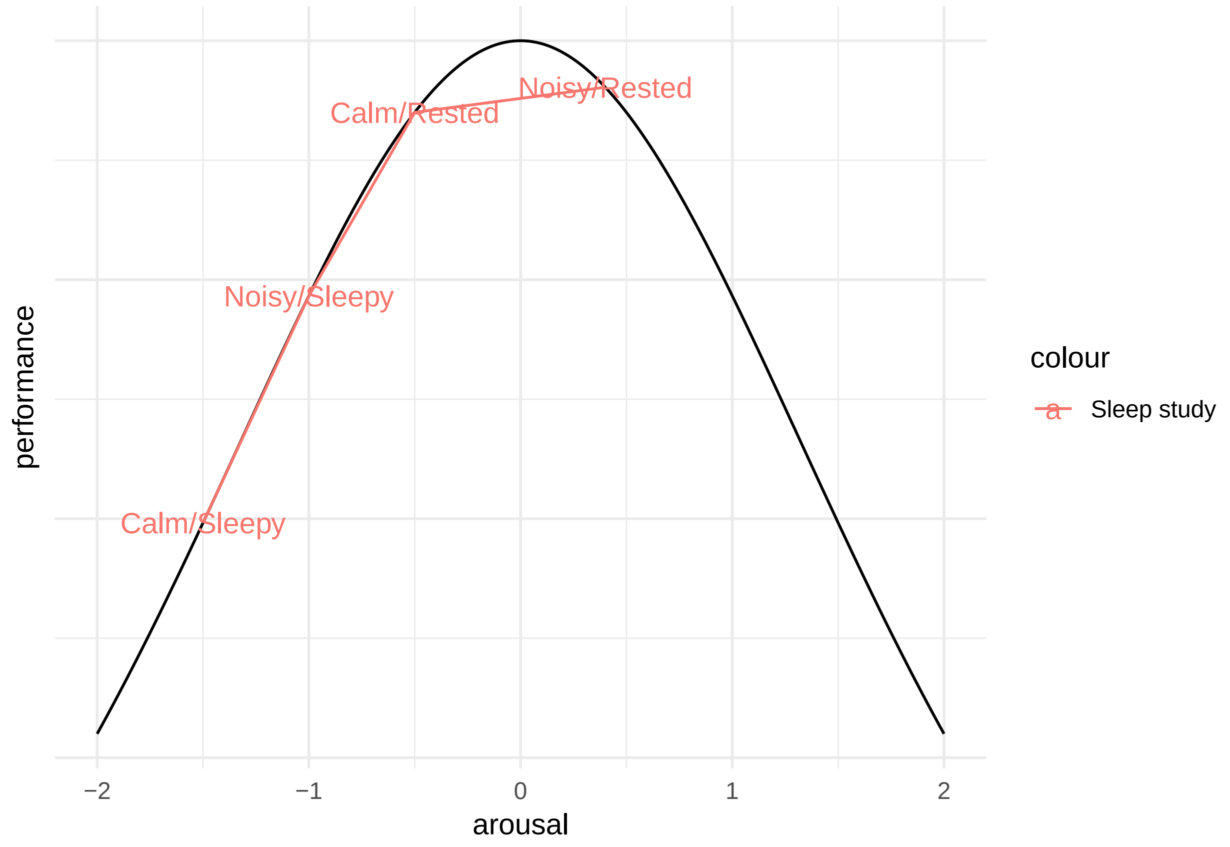
Table 36: Estimates with 95% credibility limits

fixef	center	lower	upper
Intercept	99.01	65.1	132.8
EnvironmentNoisy	-0.16	-51.3	51.3
SleepSleepy	160.31	113.5	209.5
EnvironmentNoisy:SleepSleepy	-98.91	-168.6	-29.6

Recall, that treatment contrasts were used, where all effects are given relative to the reference group quiet-rested, (intercept). The results confirm the deteriorating effect of sleepiness, although its exact impact is blurred by pronounced uncertainty $160.32[113.49, 209.55]_{CI95}$. Somewhat surprisingly, noise did not affect well-rested persons by much $-0.16[-51.33, 51.31]_{CI95}$. Note however, that we cannot conclude a null effect, as the credibility limits are wide. Maybe the lack of a clear effect is because steady white noise was used, not a disturbing tumult. The irritating effect of noise may therefor be minimal. As suggested by Corcoran, the effect of sleepiness on performance is partly reduced in a noisy environment $-98.91[-168.62, -29.62]_{CI95}$. This suggests that the arousal system is involved in the deteriorating effect of sleep deprivation, which has interesting consequences for the design of vigilance tasks in the real world.

The finding also relates to the well known Yerkes-Dodson law in ergonomic science. The law states that human performance at cognitive tasks is influenced by arousal. The influence is not linear, but better approximated with a curve as shown in Figure XY. Performance is highest at a moderate level of arousal. If we assume that sleepy participants in Corcona's study showed low performance due to under-arousal, the noise perhaps has increased the arousal level, resulting in better performance. If we except that noise has an arousing effect, the null effect of noise on rested participants stands in opposition to the Yerkes-Dodson law: if rested participants were on an optimal arousal level, additional arousal would usually have a negative effect on performance. There is the slight possibility, that Corcona has hit a sweet spot: if we assume that calm/rested participants were still below an optimal arousal level, noise could have pushed them right to the opposite point.

Central to the Yerkes-Dodson law is that arousal is a gradually increasing property, but the current experiment only features two levels. A straight line is the only way to unambiguously connect two group means; examining any curved relationships is impossible. We could think of varying noise levels over a wider range for better tracing the non-linear relationship between arousal and performance. The next section introduces polynomial regression for approximating non-linear associations between metric variables.



```r
detach(Sleep)
```

Exercises

1. In case IPump (data set D_agg), two infusion pumps haven been tested against each other (L = legacy, N = novel). Nurses from two professional groups (intensive and general care) completed a sequence of tasks with both devices. In order to account for learning effects, this procedure was repeated in three sessions. Three performance measures were taken: time-on-task, deviations from the optimal path and self-reported workload. Analyze the results using the models that have been introduced so far. Start with basic CGM and LRM, then use these as building blocks for more complex models. At least for the more basic models, always start by an exploratory plot. Then build and run the model. Extract the coefficients table and interpret magnitude and certainty.

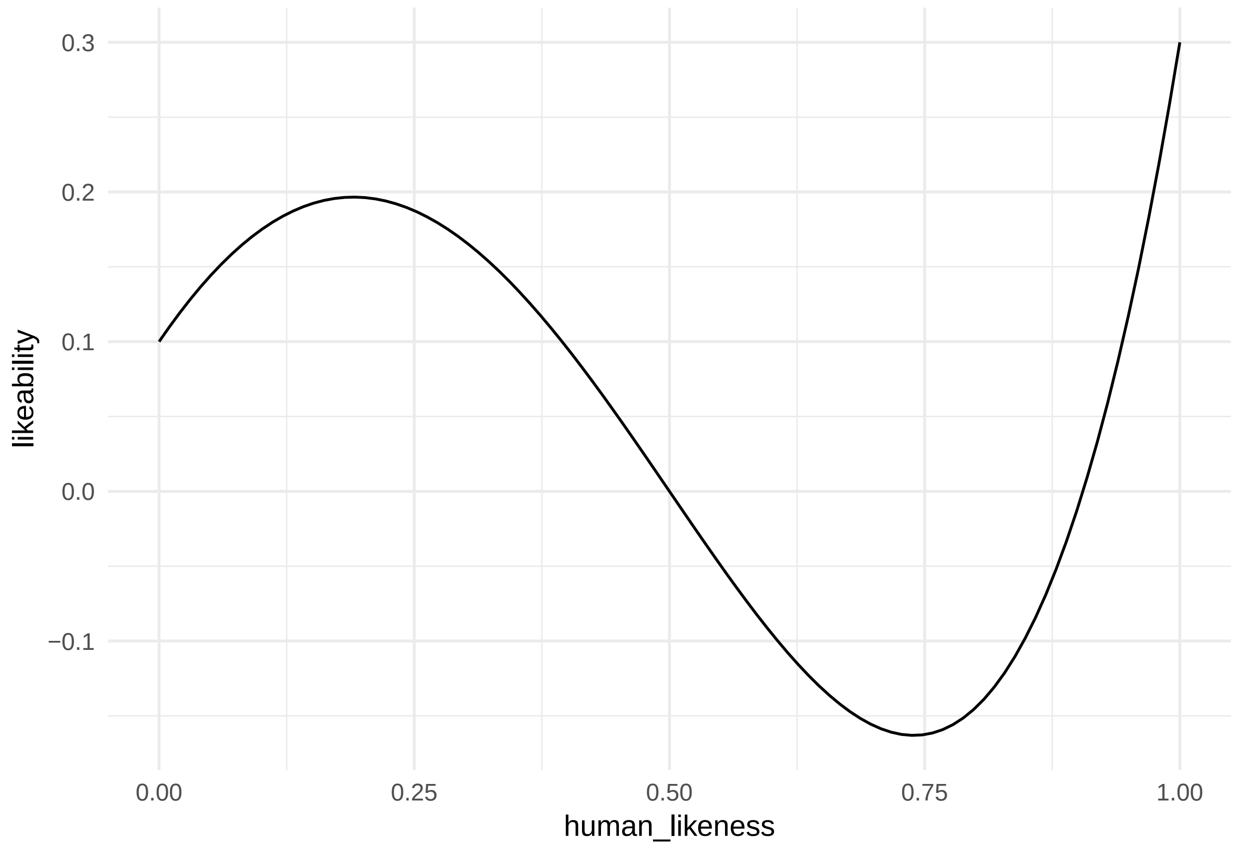
2. For the basic models of the previous exercise, extract and plot the residual distribution. Compare the residual distribution by groups.

3. For the more complex models, extract the predicted values and residuals and plot them. Is their an association between the two?

Doing the rollercoaster: polynomial regression

Robots build our cars and sometimes drive them. They mow the lawn and may soon also deliver parcels to far-off regions. Prophecy is that robots will also enter social domains, such as accompany children and care for our seniors. One can assume that in social settings emotional acceptance plays a significant role for technology adoption. Next to our voices, our faces and mimic expressions are the main source of interpersonal messaging. Since the dawn of the very idea of robots, anthropomorphic designs have been dominant. Researchers and designers all around the globe are currently pushing the limits of human-likeness of robots. One could assume that emotional response improves with every small step towards perfection. Unfortunately, this is not so. [Mori] discovered a bizarre non-linearity in human response: people respond more positive to improvements in human-likeness, but only at the lower end. A feline robot design with recognizable but stylized facial features will always beat a faceless robot with a strong mechanical appeal. Designs on the high end, that are very human-like, but not exactly, face a sudden drop in emotional response, which is called the *uncanny valley*.

```r
G_uncanny_illu <-
  data_frame(hl = seq(-1, 1, length.out = 100),
             likeability = -.5 * hl + .6 * hl^3 + .2 * hl^4) %>%
  mutate(human_likeness = (hl + 1)/2) %>%
  ggplot(aes(x = human_likeness, y = likeability)) +
  geom_line()

G_uncanny_illu
```



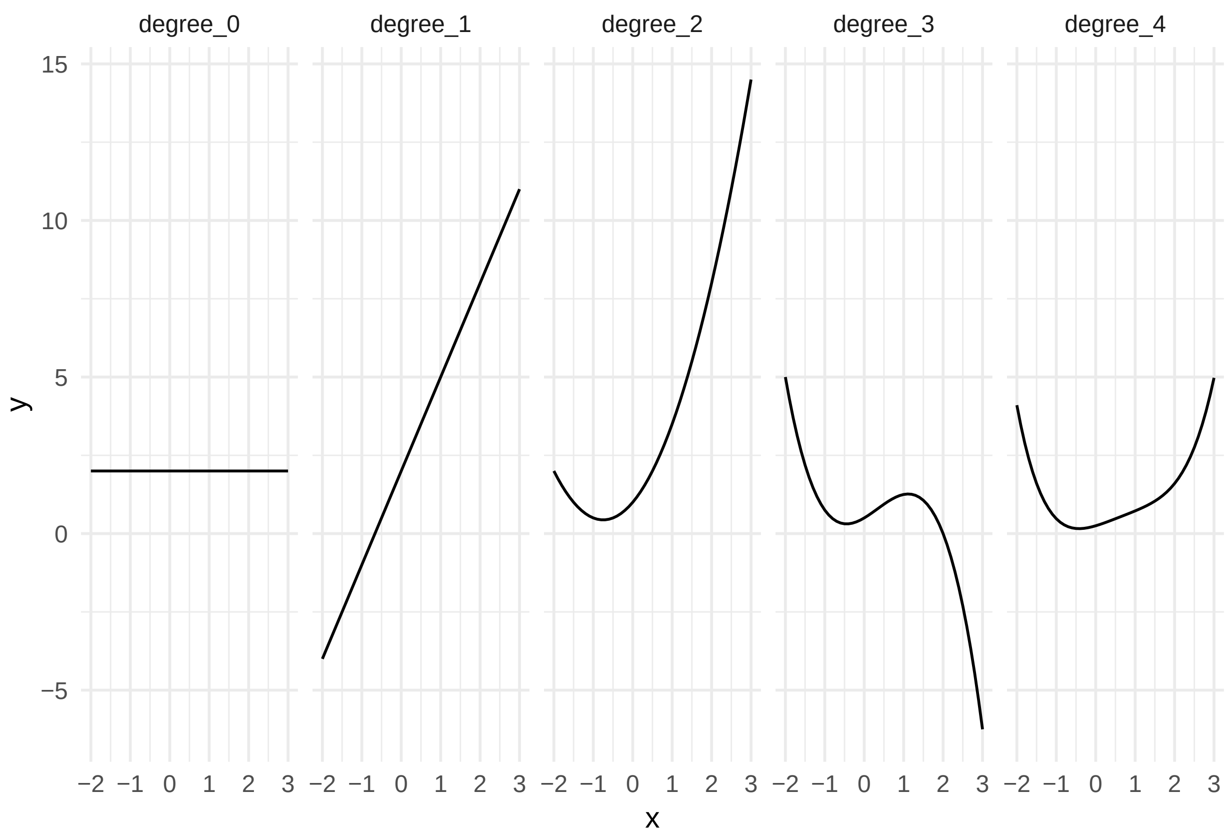
[Mathur et al.] study aimed at rendering the association between human-likeness and liking at full range. They collected 60 pictures of robots and attached a score ranging from mechanical to human appearance. Then they let more than 200 participants rate the faces on how much they liked them. Finally, they created an average score of likability per robot picture and examined the association with human likeness using a statistical model. Owing to the curved shape of the uncanny valley, linear regression is not applicable to the problem. Instead, Mathur et al. applied a third degree polynomial term.

A polynomial function of degree k has the form:

$$y_i = \beta_0 x_i^0 + \beta_1 x_i^1 + ... + \beta_k x_i^k$$

In fact, you are already familiar with two polynomial models. The zero degree polynomial is the grand mean model. This follows from $x_i^0 = 1$, which makes β_0 a constant, the intercept. Also, a first degree polynomial is just the linear model. By adding higher degrees we can introduce more complex curvature to the association.

```
D_poly <-
  data_frame(x = seq(-2, 3, by = .1),
             degree_0 = 2,
             degree_1 = degree_0 + 3 * x,
             degree_2 = 0.5 * (degree_1 + 2 * x^2),
             degree_3 = 0.5 * (degree_2 + -1 * x^3),
             degree_4 = 0.5 * (degree_3 + 0.2 * x^4)) %>%
  gather(polynomial, y, degree_0:degree_4) %>%
  arrange(polynomial, y, x)
D_poly %>%
  ggplot(aes(x, y)) +
  geom_line() +
  facet_grid(~polynomial)
```



Mathur et al. argue that the Uncanny Valley curve possesses two stationary points, where the slope is zero. One is a local minimum and represents the deepest point in the valley. The second is a local maximum and marks the shoulder left of the valley. Such a curvature can be approximated with a polynomial of (at least) third degree, which has a constant term β_0, a linear slope $x\beta_1$, quadratic component $x^2\beta_2$ and a cubic component $x^3\beta_3$.

While R provides high-level methods to deal with polynomial regression, it is instructive to build the regression manually. The first step is to add variables to the data frame, which are the exponentiated predictors ($x_k = x^k$). These variables are then straight-forwardly added to the model term, as if they were independent predictors. For better clarity, we rename the Intercept to be x_0, before summarizing the fixed effects.

```
attach(Uncanny)
```

```
M_poly <-
  UV_1 %>%
  mutate(huMech_0 = 1,
         huMech_1 = huMech,
         huMech_2 = huMech^2,
         huMech_3 = huMech^3) %>%
  stan_glm(avg_like ~  + huMech_1 + huMech_2 + huMech_3,
           data = ., iter = 400)
  # stan_glm(avg_like ~ poly(huMech, 4),
  #          data = ., iter = 200)
```

```r
P_poly <- posterior(M_poly)
```

```r
# Uncanny$P_1 %>%
#   mutate(parameter = str_replace(parameter, "Intercept", "huMech_0"),
#          fixef = str_replace(fixef, "Intercept", "huMech_0"))
```

```r
library(polynom)
```

```r
T_fixef_1 <- fixef(P_poly)
T_fixef_1
```

Table 37: Estimates with 95% credibility limits

fixef	center	lower	upper
Intercept	-0.207	-0.337	-0.093
huMech_1	-0.525	-0.763	-0.281
huMech_2	0.293	0.060	0.577
huMech_3	0.744	0.368	1.133

We can extract the fixef effects table as usual. The four coefficients specify the polynomial to approximate the average likability responses. The polynomial parameters have little explanatory value. Neither of the parameter alone relates to a relevant property of the uncanny valley. One relevant property would be the location of the deepest point of the uncanny valley, its trough. The trough is a local minimum of the curve and with polynomial techniques, we can find this point.

Finding a local minimum is a two step procedure: first, we must find all *stationary points*, which includes local minima and maxima. Then, we determine which of the resulting points is the local minimum. Stationary points occur, where the curve bends from a rising to falling, or vice versa. They are distinguishable by having a slope of zero, neither rising nor falling. Stationary points can be identified by the derivative of the third degree polynomial, which is a second degree polynomial:

$$f'(x) = \beta_1 + 2\beta_2 x + 3\beta_2 x^2$$

The derivative $f'(x)$ of a function $f(x)$ gives the slope of $f(x)$ at any given point x. When $f'(x) > 0$, $f(x)$ is rising at x, with $f'(x) < 0$ it is falling. Stationary points are precisely those points, where $f'(x) = 0$ and can be found by solving the equation. The derivative of a third degree polynomial is of the second degree, which has a quadratic part. This can produce a parabolic form, which hits point zero twice, during rise and when falling. A rising encounter of point zero indicates that $f(x)$ has a local minimum at x, a local maximum when falling. In consequence, solving $f'(x) = 0$ can result in two solutions, one minimum and one maximum, which needs to be distinguished further.

If the stationary point is a local minimum, as the trough, slope switches from negative to positive; $f'(x)$ crosses $x = 0$ in a rising manner, which is a positive slope of $f'(x)$. Therefore, a stationary point is a local minimum, when of $f''(x) > 0$.

Mathur et al. followed these analytic steps to arrive at an estimate for the position of the trough. The following code uses several high-level functions from package `polynom` to estimate the location of the trough, by drawing on the first and second derivative `d[d]poly`

```r
poly      = polynomial(T_fixef_1$center) # UC function on center
dpoly     = deriv(poly)                  # 1st derivative
ddpoly    = deriv(dpoly)                 # 2nd derivative
stat_pts = solve(dpoly)                  # finding stat points
slopes    = as.function(ddpoly)(stat_pts)# slope at stat points
trough    = stat_pts[slopes > 0]         # selecting the local minimum

cat("The trough is most likely at a huMech score of ", round(trough, 2))
```

```
## The trough is most likely at a huMech score of  0.37
```

While this procedure is instructive, there is an issue: drawing on the center estimates, which is a summary of the PD, we get a point estimate, only. Statements on certainty are impossible, as a CI is lacking. Recall the 99 seconds study, where we operated directly on the PD to obtain more specific statements on certainty. Another case for directly operating on the PD samples is to calculate additional statistics. It is another virtue of the MCMC method, that this is possible.

Every PD sample contains simultaneous draw of the four parameters `huMech_[0:3]`, and therefore fully specifies its own third degree polynomial. A PD for the trough parameter can be obtained by performing the above procedure on every sample separately. For the convenience, the case study Uncanny contains a function `trough(coef)` that includes all the above steps. The following code creates a data frame with one row per MCMC draw and the four huMech variables, the function `trough` acts on this data frame as a matrix of coefficients and returns one trough point per row. We have obtained the PD of the trough.

```r
P_trough <-
  P_poly %>%
  filter(type == "fixef") %>%
  select(chain, iter, fixef, value) %>%
  spread(fixef, value) %>%
  select(Intercept, starts_with("huMech")) %>%
  trough()
```

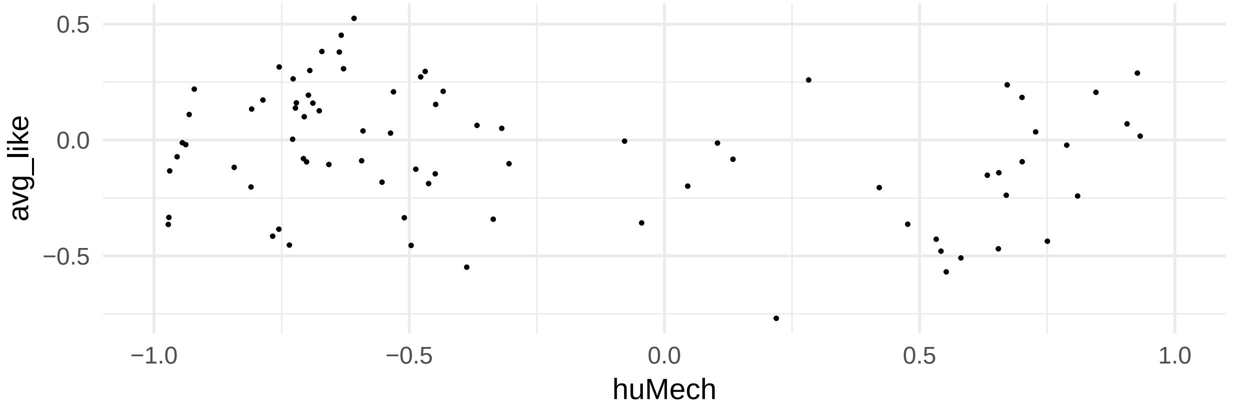
From the PD, we can derive statements on uncertainty in the usual way: the 95% credibility limits we get by:

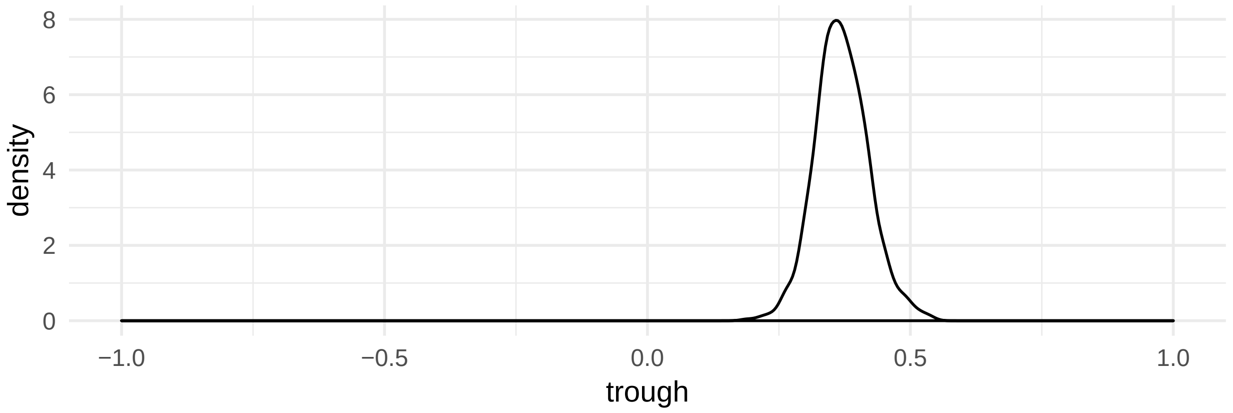
```r
quantile(P_trough, c(.025, 0.5, .975), na.rm = T) %>%
  kable()
```

	x
2.5%	0.271
50%	0.367
97.5%	0.479

The 95% CI is a conventional measure of uncertainty and may be more or less irrelevant for the spectator. The most generous account on uncertainty is a density plot on the full posterior. The density function just smooths over the frequency distribution of trough draws, but makes no arbitrary choices on where to cut it.

```r
grid.arrange(
  UV_1 %>%
    ggplot(aes(x = huMech, y = avg_like)) +
    geom_point(size = .3) +
    xlim(-1, 1),

  data_frame(trough = P_trough) %>%
    ggplot(aes(x = trough)) +
    geom_density(aes(x = trough)) +
    xlim(-1, 1)
)
```





With reasonable certainty, we can say that the trough is at approximately two-thirds of the huMech score range. The illustration of the uncanny valley as they used to be perpetuated from the original source, place the trough at about four quarters of the scale. We clearly see that this is not the case in Mathur's study. This might be an artifact, for example in that the huMech score is not linearly related to the perception of human-likeness.

```
detach(Uncanny)
```