Generalized Linear Models with `brms`

Filippo Gambarota

filippo.gambarota@unipd.it

University of Padova

Recap about linear models

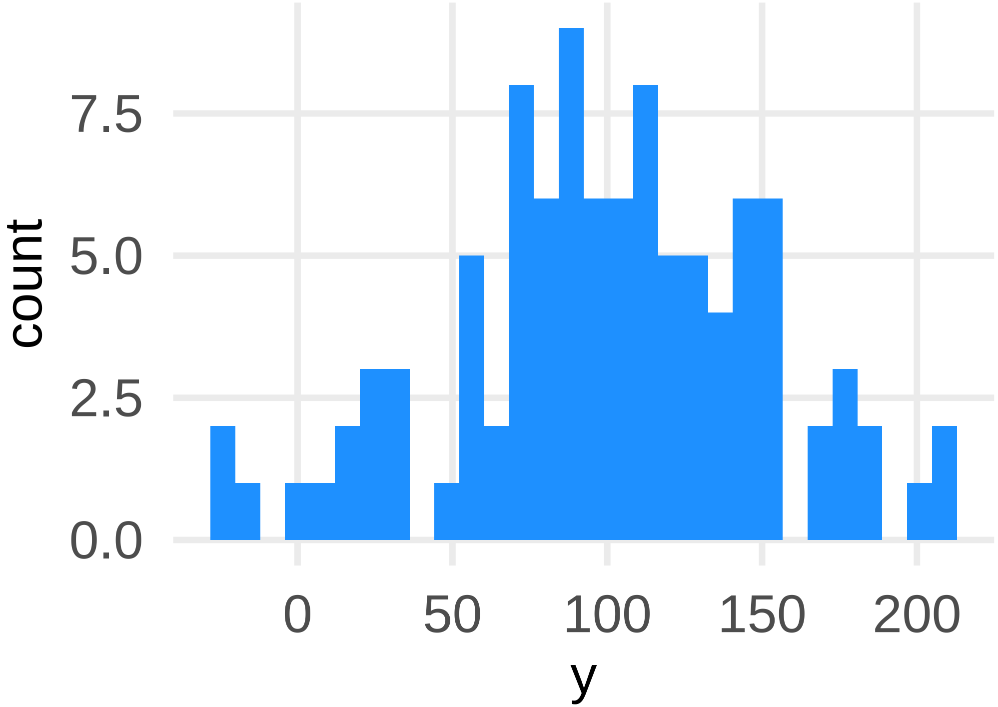
(almost) everything is a linear model

Most of the statistical analysis that you usually perfom, is essentially a linear model.

- The **t-test** is a linear model where a numerical variable `y` is predicted by a factor with two levels `x`
- The **one-way anova** is a linear model where a numerical variable `y` is predicted by one factor with more than two levels `x`
- The **correlation** is a linear model where a numerical variable `y` is predicted by another numerical variable `x`
- The **ancova** is a linear model where a numerical variable `y` is predicted by a numerical variable `x` and a factor with two levels `g`
- ...

What is a linear model?

Let's start with a single variable y. We assume that the variable comes from a Normal distribution:



What is a linear model?

What we can do with this variable? We can estimate the parameters that define the Normal distribution thus μ (the mean) and σ (the standard deviation).

```r
mean(y)
#> [1] 100
sd(y)
#> [1] 50
```

Using a linear model we can just fit a model without predictors, also known as intercept-only model.

```r
fit <- glm(y ~ 1, family = gaussian(link = "identity"))
summary(fit)
```

What is a linear model?

```
#>
#> Call:
#> glm(formula = y ~ 1, family = gaussian(link = "identity"))
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)      100          5      20   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for gaussian family taken to be 2500)
#>
#>     Null deviance: 247500  on 99  degrees of freedom
#> Residual deviance: 247500  on 99  degrees of freedom
#> AIC: 1069.2
```

What is a linear model?

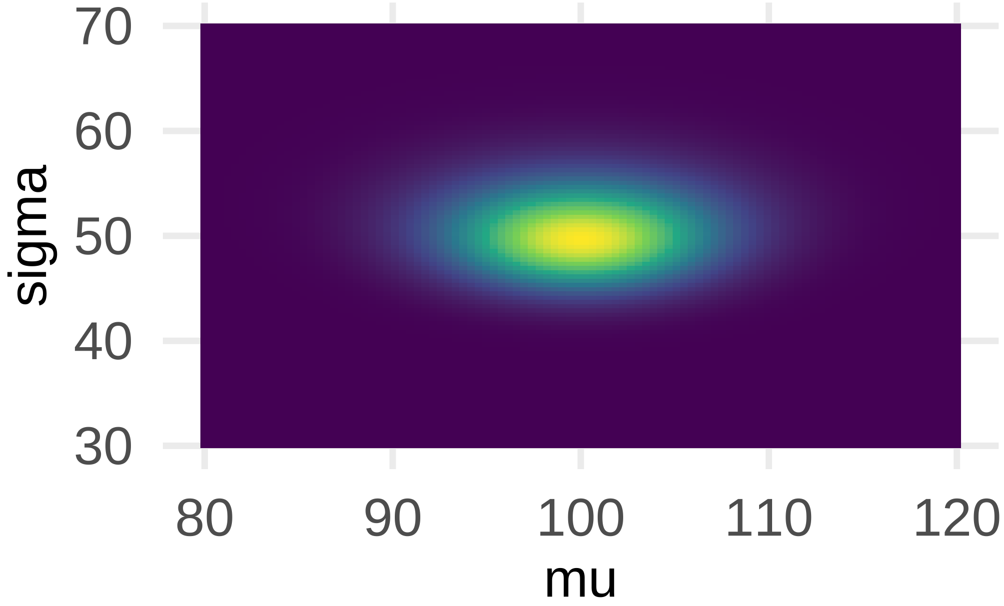
```
#>
#> Number of Fisher Scoring iterations: 2
```

I am using `glm` because I want to estimate parameters using Maximul Likelihood, but the results are the same as using `lm`.

Basically we estimated the mean `(Intercept)` and the standard deviation `Dispersion`, just take the square root thus 50.

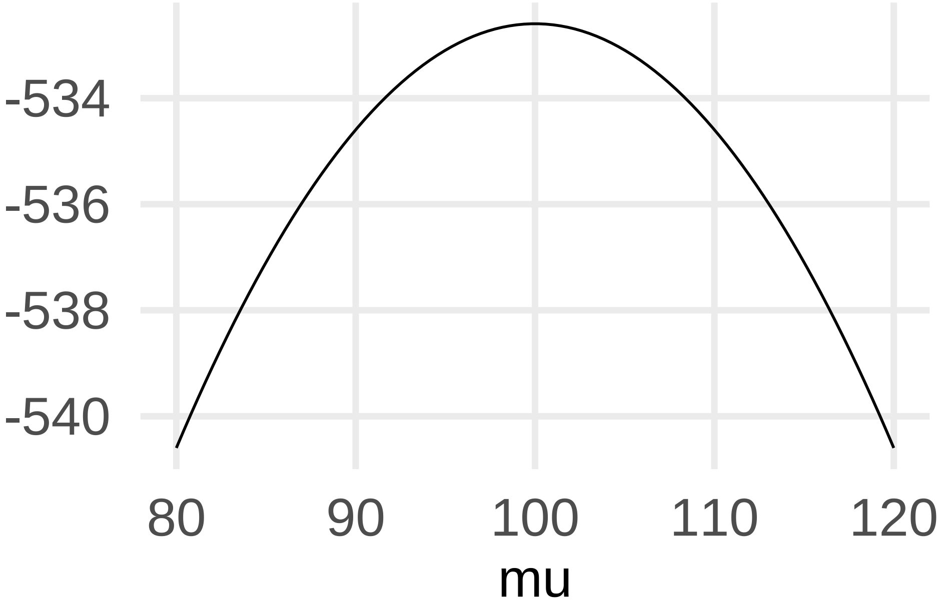
What we are doing is essentially finding the μ and σ that maximised the log-likelihood of the model fixing the observed data.

What is a linear model?



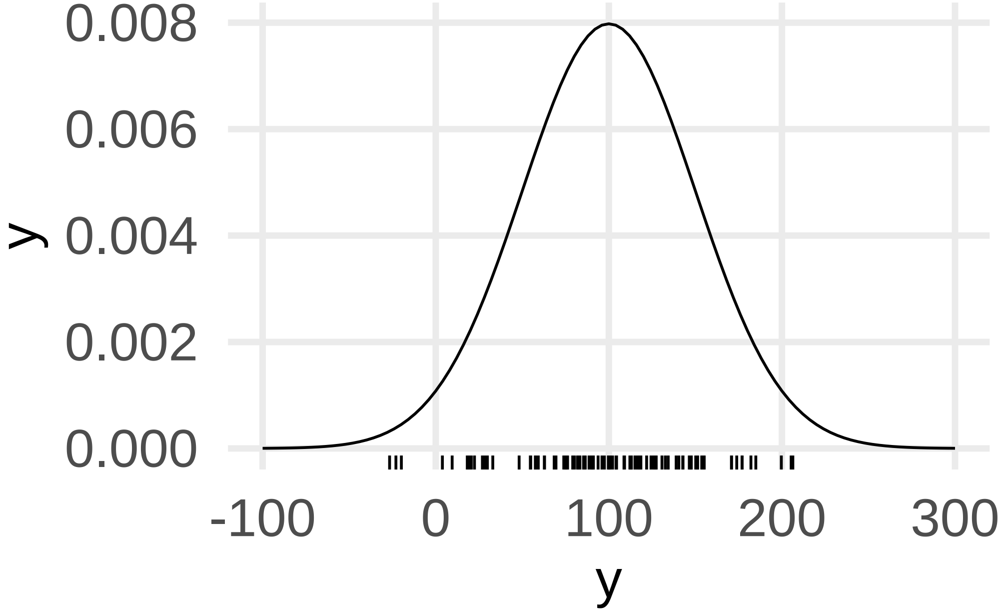
And assuming that we know σ (thus fixing it at 50):

What is a linear model?

$=$ 

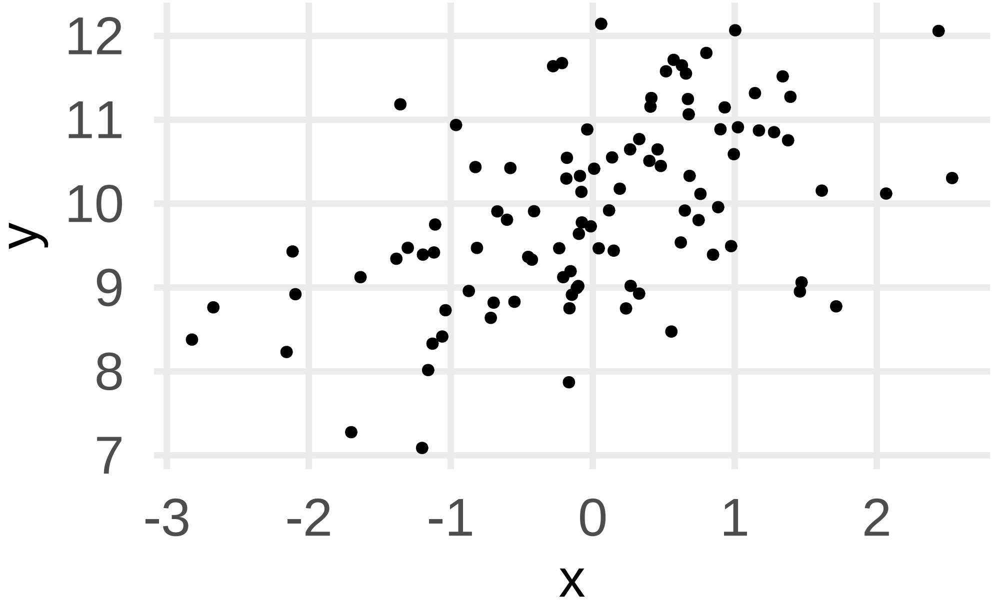
Thus, with the estimates of `glm`, we have this model fitted on the data:

What is a linear model?



Including a predictor

When we include a predictor, we are actually try to explain the variability of `y` using a variable `x`. For example, this is an hypothetical relationship:



Including a predictor

Seems that there is a positive (linear) relationship between `x` and `y`. We can try to improve the previous model by adding the predictor:

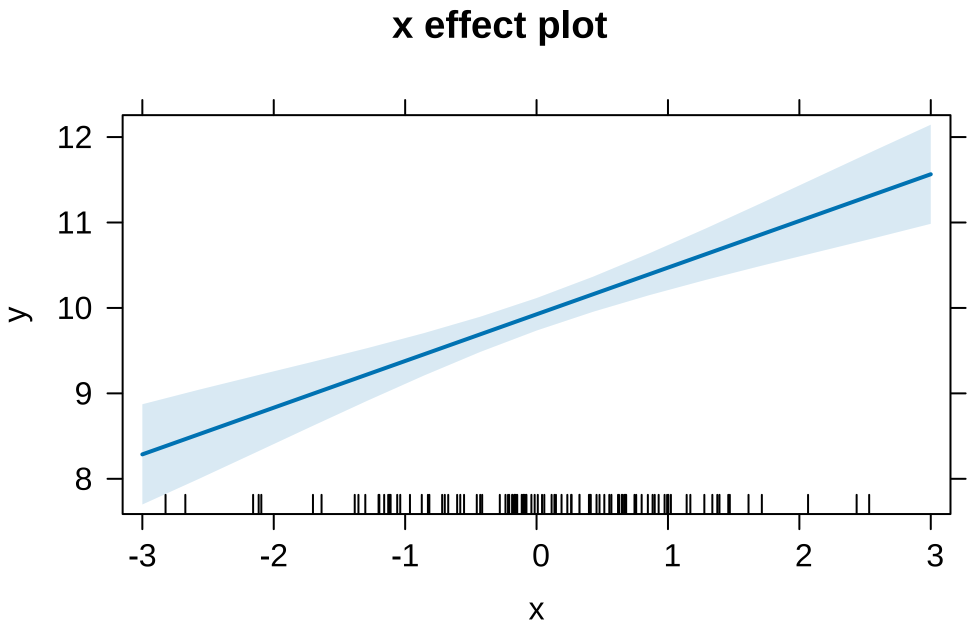
```r
fit <- glm(y ~ x, family = gaussian(link = "identity"))
summary(fit)
```

```
#> 
#> Call:
#> glm(formula = y ~ x, family = gaussian(link = "identity"))
#> 
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)    
#> (Intercept)  9.92538    0.09590 103.500  < 2e-16 ***
#> x            0.54648    0.09272   5.894 5.35e-08 ***
#> ---
```

Including a predictor

```
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for gaussian family taken to be 0.9193148)
#>
#>     Null deviance: 122.028  on 99  degrees of freedom
#> Residual deviance:  90.093  on 98  degrees of freedom
#> AIC: 279.35
#>
#> Number of Fisher Scoring iterations: 2
```

Including a predictor



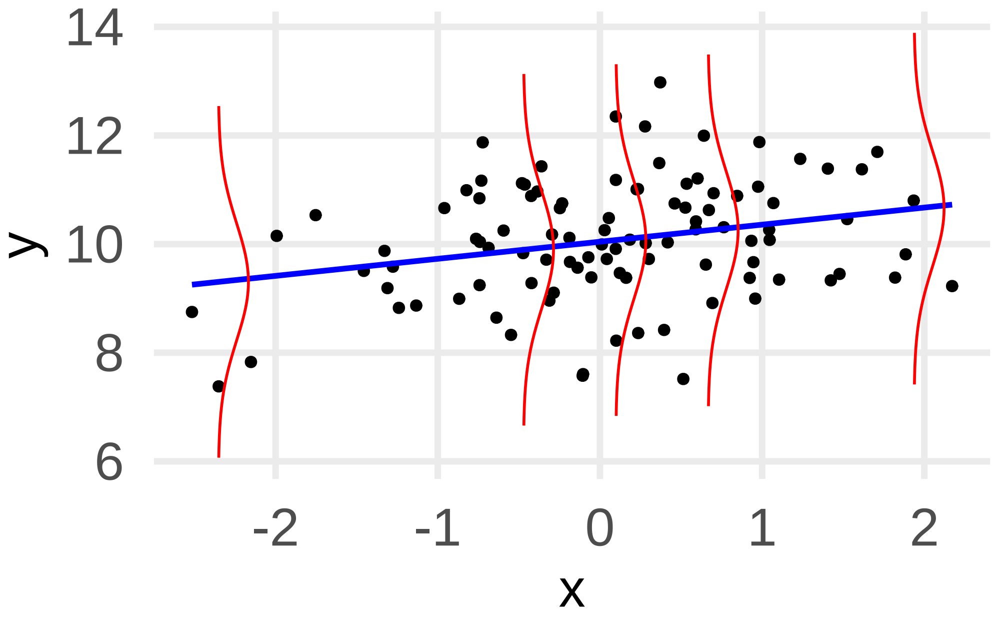
Assumptions of the linear model

Assumptions of the linear model

include here

Assumptions of the linear model

More practicaly, we are saying that the model allows for varying the mean i.e., each x value can be associated with a different μ but with a fixed (and estimated) σ.



Bayesian Models

`rstanarm`

Let's fit the same model but with `rstanarm`. I'm using `rstanarm` just because is faster, but the idea (and the result) is the same using `brms`.

```
#>
#> SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
#> Chain 1:
#> Chain 1: Gradient evaluation took 2.7e-05 seconds
#> Chain 1: 1000 transitions using 10 leapfrog steps per transition would
take 0.27 seconds.
#> Chain 1: Adjust your expectations accordingly!
#> Chain 1:
#> Chain 1:
#> Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
#> Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
#> Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
```

rstanarm

```
#> Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
#> Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
#> Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
#> Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
#> Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
#> Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
#> Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
#> Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
#> Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
#> Chain 1: 
#> Chain 1:  Elapsed Time: 0.026 seconds (Warm-up)
#> Chain 1:                0.031 seconds (Sampling)
#> Chain 1:                0.057 seconds (Total)
#> Chain 1: 
#>
```

rstanarm

```
#> SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
#> Chain 2:
#> Chain 2: Gradient evaluation took 1.2e-05 seconds
#> Chain 2: 1000 transitions using 10 leapfrog steps per transition would
take 0.12 seconds.
#> Chain 2: Adjust your expectations accordingly!
#> Chain 2:
#> Chain 2:
#> Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
#> Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
#> Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
#> Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
#> Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
#> Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
#> Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
```

rstanarm

```
#> Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
#> Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
#> Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
#> Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
#> Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
#> Chain 2: 
#> Chain 2:  Elapsed Time: 0.026 seconds (Warm-up)
#> Chain 2:                0.033 seconds (Sampling)
#> Chain 2:                0.059 seconds (Total)
#> Chain 2: 
#> 
#> SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
#> Chain 3: 
#> Chain 3: Gradient evaluation took 1e-05 seconds
#> Chain 3: 1000 transitions using 10 leapfrog steps per transition would
```

rstanarm

```
take 0.1 seconds.
#> Chain 3: Adjust your expectations accordingly!
#> Chain 3:
#> Chain 3:
#> Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
#> Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
#> Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
#> Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
#> Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
#> Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
#> Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
#> Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
#> Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
#> Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
#> Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
```

rstanarm

```
#> Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
#> Chain 3: 
#> Chain 3:  Elapsed Time: 0.025 seconds (Warm-up)
#> Chain 3:                0.031 seconds (Sampling)
#> Chain 3:                0.056 seconds (Total)
#> Chain 3: 
#> 
#> SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
#> Chain 4: 
#> Chain 4: Gradient evaluation took 1.3e-05 seconds
#> Chain 4: 1000 transitions using 10 leapfrog steps per transition would
take 0.13 seconds.
#> Chain 4: Adjust your expectations accordingly!
#> Chain 4: 
#> Chain 4: 
```

rstanarm

```
#> Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
#> Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
#> Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
#> Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
#> Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
#> Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
#> Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
#> Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
#> Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
#> Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
#> Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
#> Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
#> Chain 4:
#> Chain 4:  Elapsed Time: 0.027 seconds (Warm-up)
#> Chain 4:                0.034 seconds (Sampling)
```

rstanarm

```
#> Chain 4:                        0.061 seconds (Total)
#> Chain 4:
```

```
#>
#> Model Info:
#>  function:     stan_glm
#>  family:       gaussian [identity]
#>  formula:      y ~ x
#>  algorithm:    sampling
#>  sample:       4000 (posterior sample size)
#>  priors:       see help('prior_summary')
#>  observations: 100
#>  predictors:   2
#>
#> Estimates:
```

rstanarm

```
#>                mean   sd   10%   50%   90%
#> (Intercept) 10.0    0.1  9.9  10.0  10.2
#> x            0.3    0.1  0.2   0.3   0.5
#> sigma        1.1    0.1  1.0   1.1   1.2
#>
#> Fit Diagnostics:
#>             mean   sd   10%   50%   90%
#> mean_PPD 10.1    0.2  9.9  10.1  10.3
#>
#> The mean_ppd is the sample average posterior predictive distribution of
the outcome variable (for details see help('summary.stanreg')).
#>
#> MCMC diagnostics
#>                mcse Rhat n_eff
#> (Intercept)   0.0  1.0  3598
```

rstanarm

```
#> x             0.0  1.0  3420
#> sigma         0.0  1.0  3490
#> mean_PPD      0.0  1.0  3625
#> log-posterior 0.0  1.0  1563
#>
#> For each parameter, mcse is Monte Carlo standard error, n_eff is a crude
measure of effective sample size, and Rhat is the potential scale reduction
factor on split chains (at convergence Rhat=1).
```

```
#> # A draws_df: 1000 iterations, 4 chains, and 3 variables
#>    (Intercept)    x sigma
#> 1         10.1 0.36   1.1
#> 2         10.1 0.31   1.1
#> 3         10.0 0.34   1.1
#> 4         10.2 0.27   1.0
```

rstanarm

```
#> 5          10.1 0.42   1.0
#> 6           9.9 0.21   1.2
#> 7           9.8 0.27   1.1
#> 8          10.2 0.21   1.2
#> 9          10.2 0.36   1.1
#> 10          9.9 0.28   1.1
#> # ... with 3990 more draws
#> # ... hidden reserved variables {'.chain', '.iteration', '.draw'}
```