Model Comparison and Interpretation

```
library(pracma)
library(posterior)
```

```
This is posterior version 1.2.1


Attaching package: 'posterior'

The following objects are masked from 'package:stats':

    mad, sd, var
```

```
library(cmdstanr)
```

```
Warning: package 'cmdstanr' was built under R version 4.1.3

This is cmdstanr version 0.5.3

- CmdStanR documentation and vignettes: mc-stan.org/cmdstanr

- CmdStan path: /Users/stefano/.cmdstan/cmdstan-2.30.1

- CmdStan version: 2.30.1


A newer version of CmdStan is available. See ?install_cmdstan() to install it.
To disable this check set option or environment variable CMDSTANR_NO_VER_CHECK=TRUE.
```

```r
library(rstan)
```

```
Loading required package: StanHeaders
```

```
Loading required package: ggplot2
```

```
rstan (Version 2.21.5, GitRev: 2e1f913d3ca3)
```

```
For execution on a local, multicore CPU with excess RAM we recommend calling
options(mc.cores = parallel::detectCores()).
To avoid recompilation of unchanged Stan programs, we recommend calling
rstan_options(auto_write = TRUE)
```

```
Attaching package: 'rstan'
```

```
The following objects are masked from 'package:posterior':

    ess_bulk, ess_tail
```

```r
library(ggplot2)
library(loo)
```

```
This is loo version 2.5.1
```

```
- Online documentation and vignettes at mc-stan.org/loo
```

```
- As of v2.0.0 loo defaults to 1 core but we recommend using as many as possible. Use the 'co
```

```
Attaching package: 'loo'
```

```
The following object is masked from 'package:rstan':

    loo
```

```r
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```r
library(tidyr)
```

```
Attaching package: 'tidyr'

The following object is masked from 'package:rstan':

    extract
```

```r
library(brms)
```

```
Loading required package: Rcpp

Loading 'brms' package (version 2.18.0). Useful instructions
can be found by typing help('brms'). A more detailed introduction
to the package is available through vignette('brms_overview').


Attaching package: 'brms'

The following object is masked from 'package:rstan':

    loo
```

```
The following object is masked from 'package:posterior':

    rhat

The following object is masked from 'package:pracma':

    bernoulli

The following object is masked from 'package:stats':

    ar
```

```
library(firatheme)
library(nlmeU)
```

```
Attaching package: 'nlmeU'

The following object is masked from 'package:stats':

    sigma
```

```
library(corrplot)
```

```
corrplot 0.92 loaded
```

```
library(nlme)
```

```
Attaching package: 'nlme'

The following object is masked from 'package:dplyr':

    collapse
```

```r
library(lattice)
library(plot.matrix)
library(lme4)
```

```
Loading required package: Matrix
```

```

Attaching package: 'Matrix'
```

```
The following objects are masked from 'package:tidyr':

    expand, pack, unpack
```

```
The following objects are masked from 'package:pracma':

    expm, lu, tril, triu
```

```

Attaching package: 'lme4'
```

```
The following object is masked from 'package:nlme':

    lmList
```

```
The following object is masked from 'package:brms':

    ngrps
```

```r
library(insight)
library(firatheme)
library(purrr)
```

```

Attaching package: 'purrr'
```

```
The following object is masked from 'package:pracma':

    cross
```

```r
library(patchwork)
library(glue)
library(tidyverse)
```

```
-- Attaching packages --------------------------------------- tidyverse 1.3.1 --

v tibble  3.1.8     v stringr 1.4.0
v readr   2.1.1     v forcats 0.5.1

-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x nlme::collapse() masks dplyr::collapse()
x purrr::cross()   masks pracma::cross()
x Matrix::expand() masks tidyr::expand()
x tidyr::extract() masks rstan::extract()
x dplyr::filter()  masks stats::filter()
x dplyr::lag()     masks stats::lag()
x Matrix::pack()   masks tidyr::pack()
x Matrix::unpack() masks tidyr::unpack()
```

```r
cmdstanr::check_cmdstan_toolchain(fix = TRUE)
```

```
The C++ toolchain required for CmdStan is setup properly!
```

```r
register_knitr_engine(override = FALSE)
```

Data set-up

```r
data = read.csv("./data/race_results_view.csv")
```

```r
# Data processing
## Restricting my analysis to the period 2012-2021
data <- data %>% filter(
  position > 0,
  year > 2011
)
## convert to factors
data <- data %>% mutate(
```

```
  rider_name  = as.factor(rider_name),
  team_name  = as.factor(team_name)
)

# New variables
data <- data %>% group_by(year, sequence) %>% mutate(
  position_prop = (n() - position) / (n() - 1),
  prop_trans = (position_prop * (n() - 1) + 0.5) / n()
  ) %>%
  ungroup()
```

```
prior2 <- c(
    prior(gamma(1,1), class = sd)
   )
```

```
fit_year <- brm(
  formula = prop_trans ~ 0 + (1 | rider_name) + (1 | rider_name:year) + (1 | team_name)  +
  family  = Beta(),
  prior = prior2,
  data    = data,
  backend = "cmdstanr",
  chains  = 4,
  cores   = 6,
  warmup  = 1000,
  iter    = 3500
)
```

```
Start sampling

Running MCMC with 4 chains, at most 6 in parallel...

Chain 1 Iteration:    1 / 3500 [  0%]  (Warmup)
Chain 2 Iteration:    1 / 3500 [  0%]  (Warmup)
Chain 3 Iteration:    1 / 3500 [  0%]  (Warmup)
Chain 4 Iteration:    1 / 3500 [  0%]  (Warmup)
Chain 2 Iteration:  100 / 3500 [  2%]  (Warmup)
Chain 3 Iteration:  100 / 3500 [  2%]  (Warmup)
Chain 4 Iteration:  100 / 3500 [  2%]  (Warmup)
Chain 1 Iteration:  100 / 3500 [  2%]  (Warmup)
Chain 2 Iteration:  200 / 3500 [  5%]  (Warmup)
```

```
Chain 4 Iteration:  200 / 3500 [  5%]  (Warmup)
Chain 3 Iteration:  200 / 3500 [  5%]  (Warmup)
Chain 1 Iteration:  200 / 3500 [  5%]  (Warmup)
Chain 2 Iteration:  300 / 3500 [  8%]  (Warmup)
Chain 4 Iteration:  300 / 3500 [  8%]  (Warmup)
Chain 2 Iteration:  400 / 3500 [ 11%]  (Warmup)
Chain 3 Iteration:  300 / 3500 [  8%]  (Warmup)
Chain 4 Iteration:  400 / 3500 [ 11%]  (Warmup)
Chain 1 Iteration:  300 / 3500 [  8%]  (Warmup)
Chain 4 Iteration:  500 / 3500 [ 14%]  (Warmup)
Chain 3 Iteration:  400 / 3500 [ 11%]  (Warmup)
Chain 1 Iteration:  400 / 3500 [ 11%]  (Warmup)
Chain 2 Iteration:  500 / 3500 [ 14%]  (Warmup)
Chain 4 Iteration:  600 / 3500 [ 17%]  (Warmup)
Chain 1 Iteration:  500 / 3500 [ 14%]  (Warmup)
Chain 2 Iteration:  600 / 3500 [ 17%]  (Warmup)
Chain 4 Iteration:  700 / 3500 [ 20%]  (Warmup)
Chain 3 Iteration:  500 / 3500 [ 14%]  (Warmup)
Chain 1 Iteration:  600 / 3500 [ 17%]  (Warmup)
Chain 4 Iteration:  800 / 3500 [ 22%]  (Warmup)
Chain 2 Iteration:  700 / 3500 [ 20%]  (Warmup)
Chain 3 Iteration:  600 / 3500 [ 17%]  (Warmup)
Chain 1 Iteration:  700 / 3500 [ 20%]  (Warmup)
Chain 4 Iteration:  900 / 3500 [ 25%]  (Warmup)
Chain 1 Iteration:  800 / 3500 [ 22%]  (Warmup)
Chain 3 Iteration:  700 / 3500 [ 20%]  (Warmup)
Chain 2 Iteration:  800 / 3500 [ 22%]  (Warmup)
Chain 4 Iteration: 1000 / 3500 [ 28%]  (Warmup)
Chain 4 Iteration: 1001 / 3500 [ 28%]  (Sampling)
Chain 1 Iteration:  900 / 3500 [ 25%]  (Warmup)
Chain 3 Iteration:  800 / 3500 [ 22%]  (Warmup)
Chain 4 Iteration: 1100 / 3500 [ 31%]  (Sampling)
Chain 2 Iteration:  900 / 3500 [ 25%]  (Warmup)
Chain 1 Iteration: 1000 / 3500 [ 28%]  (Warmup)
Chain 1 Iteration: 1001 / 3500 [ 28%]  (Sampling)
Chain 3 Iteration:  900 / 3500 [ 25%]  (Warmup)
Chain 4 Iteration: 1200 / 3500 [ 34%]  (Sampling)
Chain 1 Iteration: 1100 / 3500 [ 31%]  (Sampling)
Chain 2 Iteration: 1000 / 3500 [ 28%]  (Warmup)
Chain 4 Iteration: 1300 / 3500 [ 37%]  (Sampling)
Chain 2 Iteration: 1001 / 3500 [ 28%]  (Sampling)
Chain 3 Iteration: 1000 / 3500 [ 28%]  (Warmup)
Chain 3 Iteration: 1001 / 3500 [ 28%]  (Sampling)
```

```
Chain 1 Iteration: 1200 / 3500 [ 34%]  (Sampling)
Chain 2 Iteration: 1100 / 3500 [ 31%]  (Sampling)
Chain 4 Iteration: 1400 / 3500 [ 40%]  (Sampling)
Chain 3 Iteration: 1100 / 3500 [ 31%]  (Sampling)
Chain 1 Iteration: 1300 / 3500 [ 37%]  (Sampling)
Chain 2 Iteration: 1200 / 3500 [ 34%]  (Sampling)
Chain 4 Iteration: 1500 / 3500 [ 42%]  (Sampling)
Chain 3 Iteration: 1200 / 3500 [ 34%]  (Sampling)
Chain 1 Iteration: 1400 / 3500 [ 40%]  (Sampling)
Chain 2 Iteration: 1300 / 3500 [ 37%]  (Sampling)
Chain 4 Iteration: 1600 / 3500 [ 45%]  (Sampling)
Chain 3 Iteration: 1300 / 3500 [ 37%]  (Sampling)
Chain 1 Iteration: 1500 / 3500 [ 42%]  (Sampling)
Chain 2 Iteration: 1400 / 3500 [ 40%]  (Sampling)
Chain 4 Iteration: 1700 / 3500 [ 48%]  (Sampling)
Chain 3 Iteration: 1400 / 3500 [ 40%]  (Sampling)
Chain 1 Iteration: 1600 / 3500 [ 45%]  (Sampling)
Chain 2 Iteration: 1500 / 3500 [ 42%]  (Sampling)
Chain 4 Iteration: 1800 / 3500 [ 51%]  (Sampling)
Chain 3 Iteration: 1500 / 3500 [ 42%]  (Sampling)
Chain 1 Iteration: 1700 / 3500 [ 48%]  (Sampling)
Chain 2 Iteration: 1600 / 3500 [ 45%]  (Sampling)
Chain 4 Iteration: 1900 / 3500 [ 54%]  (Sampling)
Chain 3 Iteration: 1600 / 3500 [ 45%]  (Sampling)
Chain 1 Iteration: 1800 / 3500 [ 51%]  (Sampling)
Chain 2 Iteration: 1700 / 3500 [ 48%]  (Sampling)
Chain 3 Iteration: 1700 / 3500 [ 48%]  (Sampling)
Chain 4 Iteration: 2000 / 3500 [ 57%]  (Sampling)
Chain 1 Iteration: 1900 / 3500 [ 54%]  (Sampling)
Chain 2 Iteration: 1800 / 3500 [ 51%]  (Sampling)
Chain 3 Iteration: 1800 / 3500 [ 51%]  (Sampling)
Chain 4 Iteration: 2100 / 3500 [ 60%]  (Sampling)
Chain 1 Iteration: 2000 / 3500 [ 57%]  (Sampling)
Chain 2 Iteration: 1900 / 3500 [ 54%]  (Sampling)
Chain 3 Iteration: 1900 / 3500 [ 54%]  (Sampling)
Chain 4 Iteration: 2200 / 3500 [ 62%]  (Sampling)
Chain 1 Iteration: 2100 / 3500 [ 60%]  (Sampling)
Chain 2 Iteration: 2000 / 3500 [ 57%]  (Sampling)
Chain 3 Iteration: 2000 / 3500 [ 57%]  (Sampling)
Chain 4 Iteration: 2300 / 3500 [ 65%]  (Sampling)
Chain 1 Iteration: 2200 / 3500 [ 62%]  (Sampling)
Chain 2 Iteration: 2100 / 3500 [ 60%]  (Sampling)
Chain 3 Iteration: 2100 / 3500 [ 60%]  (Sampling)
```

```
Chain 4 Iteration: 2400 / 3500 [ 68%]  (Sampling)
Chain 1 Iteration: 2300 / 3500 [ 65%]  (Sampling)
Chain 2 Iteration: 2200 / 3500 [ 62%]  (Sampling)
Chain 3 Iteration: 2200 / 3500 [ 62%]  (Sampling)
Chain 4 Iteration: 2500 / 3500 [ 71%]  (Sampling)
Chain 1 Iteration: 2400 / 3500 [ 68%]  (Sampling)
Chain 2 Iteration: 2300 / 3500 [ 65%]  (Sampling)
Chain 3 Iteration: 2300 / 3500 [ 65%]  (Sampling)
Chain 4 Iteration: 2600 / 3500 [ 74%]  (Sampling)
Chain 1 Iteration: 2500 / 3500 [ 71%]  (Sampling)
Chain 2 Iteration: 2400 / 3500 [ 68%]  (Sampling)
Chain 3 Iteration: 2400 / 3500 [ 68%]  (Sampling)
Chain 4 Iteration: 2700 / 3500 [ 77%]  (Sampling)
Chain 1 Iteration: 2600 / 3500 [ 74%]  (Sampling)
Chain 2 Iteration: 2500 / 3500 [ 71%]  (Sampling)
Chain 3 Iteration: 2500 / 3500 [ 71%]  (Sampling)
Chain 4 Iteration: 2800 / 3500 [ 80%]  (Sampling)
Chain 1 Iteration: 2700 / 3500 [ 77%]  (Sampling)
Chain 2 Iteration: 2600 / 3500 [ 74%]  (Sampling)
Chain 3 Iteration: 2600 / 3500 [ 74%]  (Sampling)
Chain 4 Iteration: 2900 / 3500 [ 82%]  (Sampling)
Chain 1 Iteration: 2800 / 3500 [ 80%]  (Sampling)
Chain 2 Iteration: 2700 / 3500 [ 77%]  (Sampling)
Chain 3 Iteration: 2700 / 3500 [ 77%]  (Sampling)
Chain 4 Iteration: 3000 / 3500 [ 85%]  (Sampling)
Chain 1 Iteration: 2900 / 3500 [ 82%]  (Sampling)
Chain 2 Iteration: 2800 / 3500 [ 80%]  (Sampling)
Chain 3 Iteration: 2800 / 3500 [ 80%]  (Sampling)
Chain 4 Iteration: 3100 / 3500 [ 88%]  (Sampling)
Chain 1 Iteration: 3000 / 3500 [ 85%]  (Sampling)
Chain 2 Iteration: 2900 / 3500 [ 82%]  (Sampling)
Chain 3 Iteration: 2900 / 3500 [ 82%]  (Sampling)
Chain 4 Iteration: 3200 / 3500 [ 91%]  (Sampling)
Chain 1 Iteration: 3100 / 3500 [ 88%]  (Sampling)
Chain 2 Iteration: 3000 / 3500 [ 85%]  (Sampling)
Chain 3 Iteration: 3000 / 3500 [ 85%]  (Sampling)
Chain 4 Iteration: 3300 / 3500 [ 94%]  (Sampling)
Chain 1 Iteration: 3200 / 3500 [ 91%]  (Sampling)
Chain 2 Iteration: 3100 / 3500 [ 88%]  (Sampling)
Chain 3 Iteration: 3100 / 3500 [ 88%]  (Sampling)
Chain 4 Iteration: 3400 / 3500 [ 97%]  (Sampling)
Chain 1 Iteration: 3300 / 3500 [ 94%]  (Sampling)
Chain 2 Iteration: 3200 / 3500 [ 91%]  (Sampling)
```

```
Chain 3 Iteration: 3200 / 3500 [ 91%]  (Sampling)
Chain 4 Iteration: 3500 / 3500 [100%]  (Sampling)
Chain 4 finished in 183.4 seconds.
Chain 1 Iteration: 3400 / 3500 [ 97%]  (Sampling)
Chain 2 Iteration: 3300 / 3500 [ 94%]  (Sampling)
Chain 3 Iteration: 3300 / 3500 [ 94%]  (Sampling)
Chain 1 Iteration: 3500 / 3500 [100%]  (Sampling)
Chain 1 finished in 189.3 seconds.
Chain 2 Iteration: 3400 / 3500 [ 97%]  (Sampling)
Chain 3 Iteration: 3400 / 3500 [ 97%]  (Sampling)
Chain 2 Iteration: 3500 / 3500 [100%]  (Sampling)
Chain 2 finished in 195.8 seconds.
Chain 3 Iteration: 3500 / 3500 [100%]  (Sampling)
Chain 3 finished in 196.1 seconds.

All 4 chains finished successfully.
Mean chain execution time: 191.1 seconds.
Total execution time: 196.3 seconds.


Warning: 34 of 10000 (0.0%) transitions ended with a divergence.
See https://mc-stan.org/misc/warnings for details.
```

Model fits

```
#fit_basic = readRDS("./fit/fit_basic.rds")
#summary(fit_basic)
```

```
#fit_year = readRDS("./fit/fit_year.rds")
summary(fit_year)
```

```
Warning: There were 34 divergent transitions after warmup. Increasing
adapt_delta above 0.8 may help. See http://mc-stan.org/misc/
warnings.html#divergent-transitions-after-warmup

 Family: beta
  Links: mu = logit; phi = identity
Formula: prop_trans ~ 0 + (1 | rider_name) + (1 | rider_name:year) + (1 | team_name) + (1 | t
   Data: data (Number of observations: 3184)
  Draws: 4 chains, each with iter = 3500; warmup = 1000; thin = 1;
```

```
         total post-warmup draws = 10000

Group-Level Effects:
~rider_name (Number of levels: 89)
              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(Intercept)     0.99      0.12     0.76     1.25 1.00     2167     3825

~rider_name:year (Number of levels: 298)
              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(Intercept)     0.45      0.05     0.37     0.54 1.00     1580     3817

~team_name (Number of levels: 74)
              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(Intercept)     0.51      0.09     0.36     0.70 1.00     2777     4593

~team_name:year (Number of levels: 158)
              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(Intercept)     0.20      0.09     0.02     0.35 1.00      763     1615

Family Specific Parameters:
    Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
phi     5.97      0.15     5.69     6.28 1.00    13519     6678

Draws were sampled using sample(hmc). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

Inference about Rider skills

```
riders_focus <- c("Rossi","Crutchlow","Marquez")
rider_mean <- as_draws_df(fit_year) %>% select(-.chain, -.iteration) %>% select(contains("
```

```
Warning: Dropping 'draws_df' class as required metadata was removed.
```

```
rider_form <- as_draws_df(fit_year) %>% select(-.chain,-.iteration) %>% select(contains("r
```

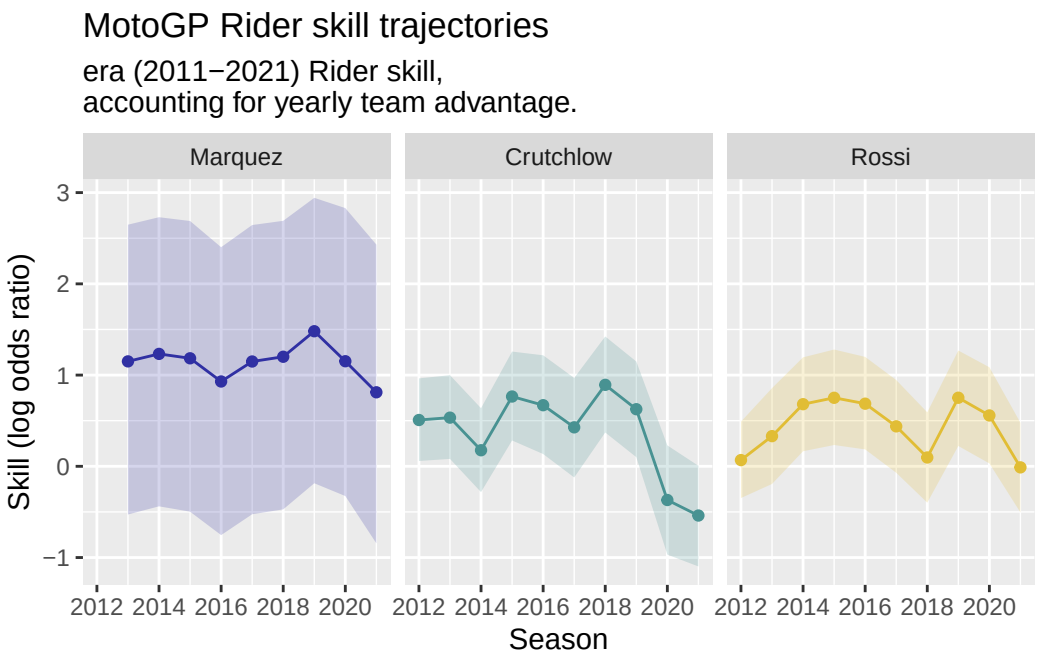
```
Warning: Dropping 'draws_df' class as required metadata was removed.
```

```
rider_mean_long <-
  rider_mean  %>%
  pivot_longer(-.draw, names_to = "Rider", values_to = "Skill",
               names_pattern = "\\[(\\w{1,10}.*?),..*?\\]") %>%
  mutate(Rider = as.factor(Rider))

rider_form_long <-
  rider_form %>%
  pivot_longer(-.draw, names_to = c("Rider", "Year"), values_to = "Form",
               names_pattern = "\\[(\\w{1,10}).*?(\\d{1,4}).*?,") %>%
  mutate(Rider = as.factor(Rider), Year = as.integer(Year))

rider_skill_summary <-
  merge(x=rider_form_long, y=rider_mean_long, by = c("Rider",".draw")) %>%
  mutate(skill_yr = Form + Skill) %>%
  group_by(Rider, Year) %>%
  summarise(
    est = mean(skill_yr),
    lower = quantile(skill_yr, 0.055),
    upper = quantile(skill_yr, 0.945),
  )
```

```
`summarise()` has grouped output by 'Rider'. You can override using the
`.groups` argument.
```

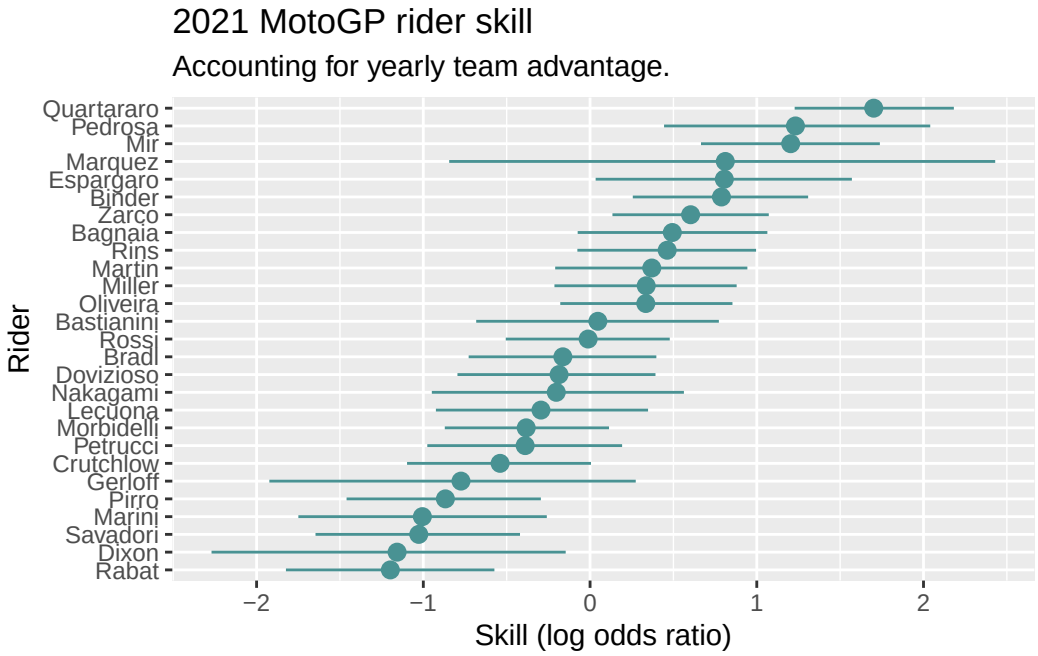
```
plt_skill_trajectory <-
  rider_skill_summary %>%
  ungroup() %>%
  filter(Rider %in% riders_focus) %>%
  mutate(Rider = fct_reorder(Rider, -est)) %>%
  ggplot(aes(x = Year, y = est, ymin = lower, ymax = upper)) +
  geom_ribbon(aes(fill = Rider), alpha = .2) +
  geom_line(aes(colour = Rider)) +
  geom_point(aes(colour = Rider)) +
  scale_fill_fira(guide = "none") +
  scale_colour_fira(guide = "none") +
  #theme_fira() +
  facet_wrap(~Rider) +
  labs(x = "Season", y = "Skill (log odds ratio)", title = "MotoGP Rider skill trajectorie
       subtitle = "era (2011-2021) Rider skill,\naccounting for yearly team advantage.")
```

```
plt_skill_trajectory
```




```
plt_rider_skill_2021 <-
  rider_skill_summary %>%
  ungroup() %>%
  filter(Year == 2021) %>%
  mutate(Rider = fct_reorder(Rider, est)) %>%
  ggplot(aes(y = Rider, x = est, xmin = lower, xmax = upper)) +
  geom_pointrange(colour = firaCols[3]) +
  #theme_fira() +
  labs(title = "2021 MotoGP rider skill",
       subtitle = "Accounting for yearly team advantage.",
       x = "Skill (log odds ratio)",
       y = "Rider")
```

```
plt_rider_skill_2021
```




```
sfit <- summary(fit_year, prob = 0.89)
```

```
Warning: There were 34 divergent transitions after warmup. Increasing
adapt_delta above 0.8 may help. See http://mc-stan.org/misc/
warnings.html#divergent-transitions-after-warmup
```

```
ranef_summary <- rbind(
  "team" = sfit$random$team_name,
  "team form" = sfit$random$`team_name:year`,
  "rider" = sfit$random$rider_name,
  "rider form" = sfit$random$`rider_name:year`
)[1:4, 1:4]
xtable::xtable(ranef_summary)
```

```
% latex table generated in R 4.1.2 by xtable 1.8-4 package
% Thu Dec  1 19:41:38 2022
\begin{table}[ht]
\centering
\begin{tabular}{rrrrr}
  \hline
```

```
 & Estimate & Est.Error & l-89\% CI & u-89\% CI \\
  \hline
team & 0.51 & 0.09 & 0.38 & 0.66 \\
  team form & 0.20 & 0.09 & 0.04 & 0.32 \\
  rider & 0.99 & 0.12 & 0.80 & 1.20 \\
  rider form & 0.45 & 0.05 & 0.38 & 0.52 \\
   \hline
\end{tabular}
\end{table}
```

```
# how much of variance is due to car?
colSums(ranef_summary[1:2,]^2)/colSums(ranef_summary^2)
```

```
 Estimate Est.Error  l-89% CI  u-89% CI
0.2033896 0.4660817 0.1592090 0.2414985
```

```
# and how much due to the driver?
colSums(ranef_summary[3:4,]^2)/colSums(ranef_summary^2)
```

```
 Estimate Est.Error  l-89% CI  u-89% CI
0.7966104 0.5339183 0.8407910 0.7585015
```