The pgainsim users guide

Nora Scherer, Pascal Schlosser

Introduction

The pgainsim package allows after performing genome-wide association studies under an additive, recessive and dominant model for a quantitative trait to determine study specific critical values when one of the three models is more informative than the other ones for a quantitative trait locus. First, study specific p-gain values are simulated. Second, based on the simulated values quantiles of the empirical density of the p-gain are computed. Finally, the quantiles are exported or interpolated and the critical values are obtained.

Installation

To install this package from the github page of the Institute of Genetic Epidemiology - University of Freiburg, start R (version “4.0”) and enter:

```
if (!requireNamespace("devtools", quietly = TRUE))
   install.packages("devtools")
if (!requireNamespace("rmarkdown", quietly = TRUE))
   install.packages("rmarkdown")
if (!requireNamespace("pgainsim", quietly = TRUE))
  devtools::install_github("genepi-freiburg/pgainsim",build_vignettes = TRUE)
```

Example Code

The function `p_gain_simulation()` provides a dataset of simulated p-gains of different modes using `pgain_types` (rec, dom, add), `AFs` (vector with different allele frequencies), `n_study` (study size), `n` (number of random draws) as input while parallizing across `cores` (number of CPU cores):

```
require(pgainsim)
```

```
## Loading required package: pgainsim
```

```
## Loading required package: parallel
```

```
## Loading required package: minpack.lm
```

```
## Loading required package: ggplot2
```

```
## Loading required package: reshape2
```

```
sim_dat <- p_gain_simulation(pgain_types=c("add","rec"), AFs=c(0.3,0.5,0.7,0.9), n=
10000L, cores=1L)
```

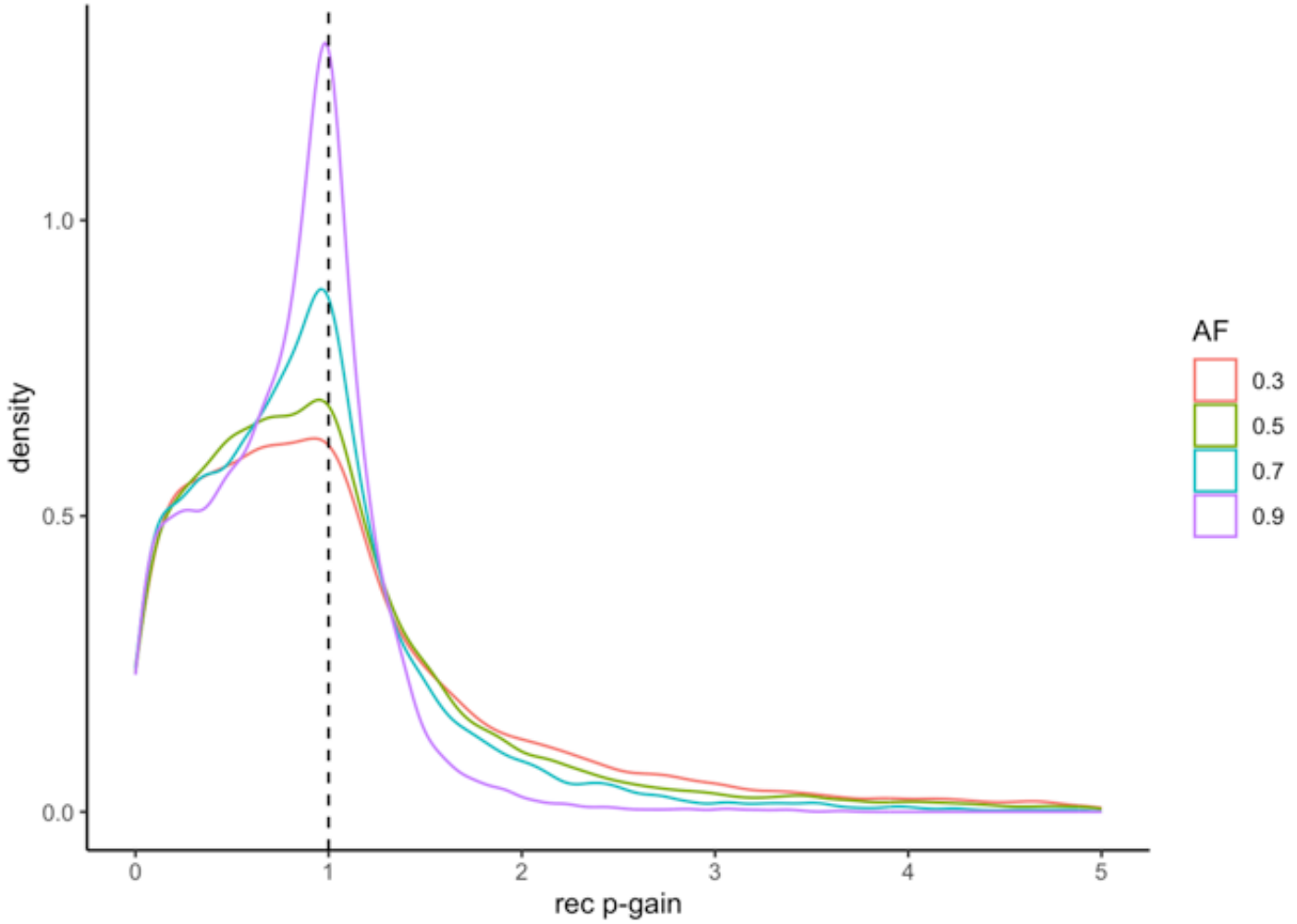
```
## [1] "Generating 10000 random draws."
```

Because the distribution of the additive p-gain is the same for the allele frequencies AF and 1-AF in the following functions the simulations for these two allele frequencies are combined for the additive p-gain.

Based on the simulated data, the empirical density of one p-gain type for different allele frequencies can be plotted by the function `p_gain_density_plot()` where `xlim` describes the range of the x-axis. Here we predefined the xlim range to zoom in on the relevant part of the density, which in turn leads to a warning as a number of data points is excluded from the plot.

```
p_gain_density_plot(pgain_type="rec", sim_data=sim_dat, xlim=c(0,5), print_pdf=FALS
E)
```

```
## Warning: Removed 730 rows containing non-finite values (`stat_density()`).
```



Based on the simulated data, the (1-0.05/#tests)-quantiles of the empirical density of the p-gain are computed by the function `p_gain_quantiles()` where #tests is the number of parallel tests and `n_tests` describes the maximal number of parallel tests:

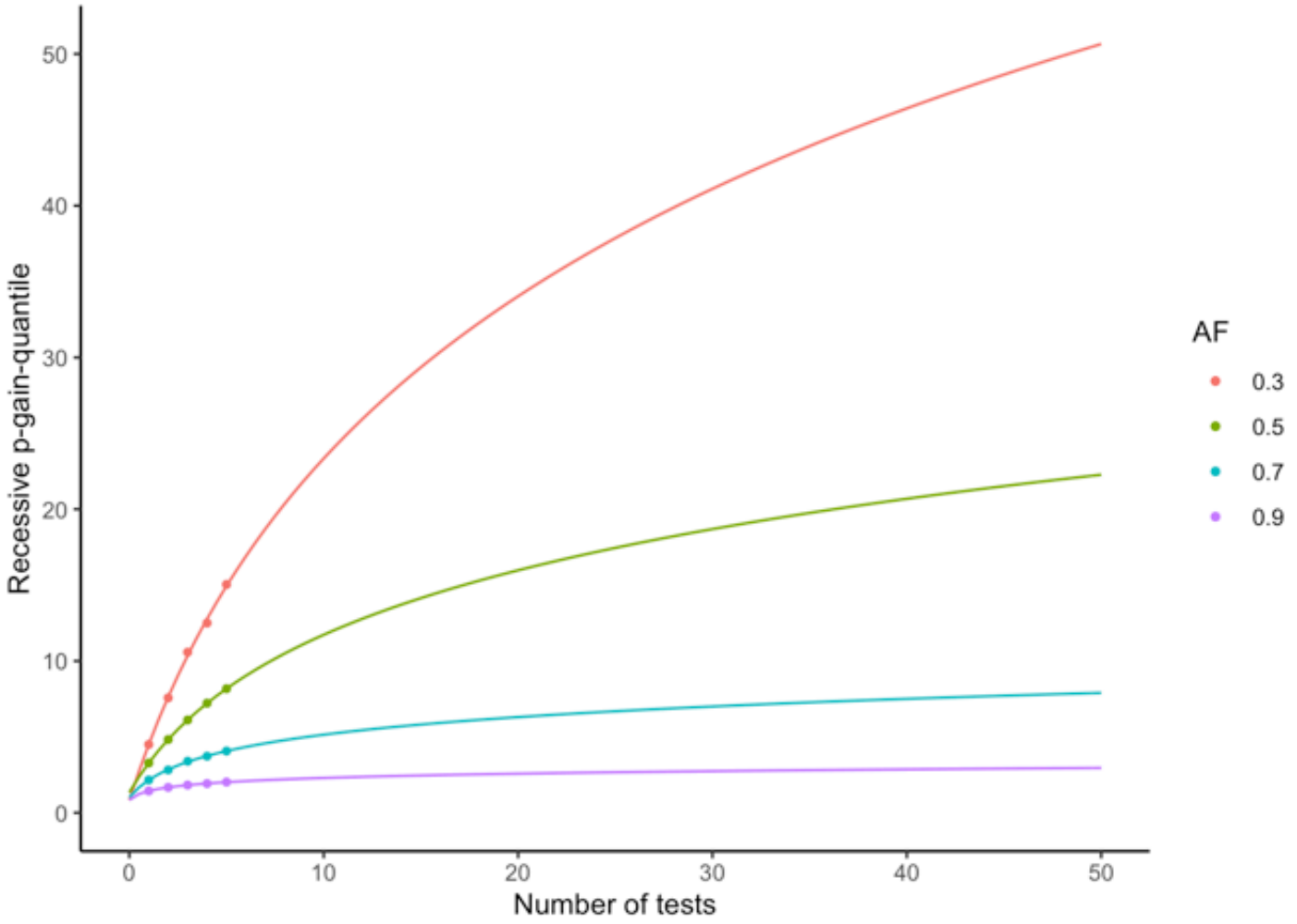
```
quantile <- p_gain_quantiles(n_tests=5L, sim_data=sim_dat)
```

For applications with extensive multiple testing the p-gain-quantiles can be interpolated by a log-linear fit by use of the function `p_gain_quantile_fit()` where `test_number` describes the number of parallel tests for which the p-gain threshold should be determined:

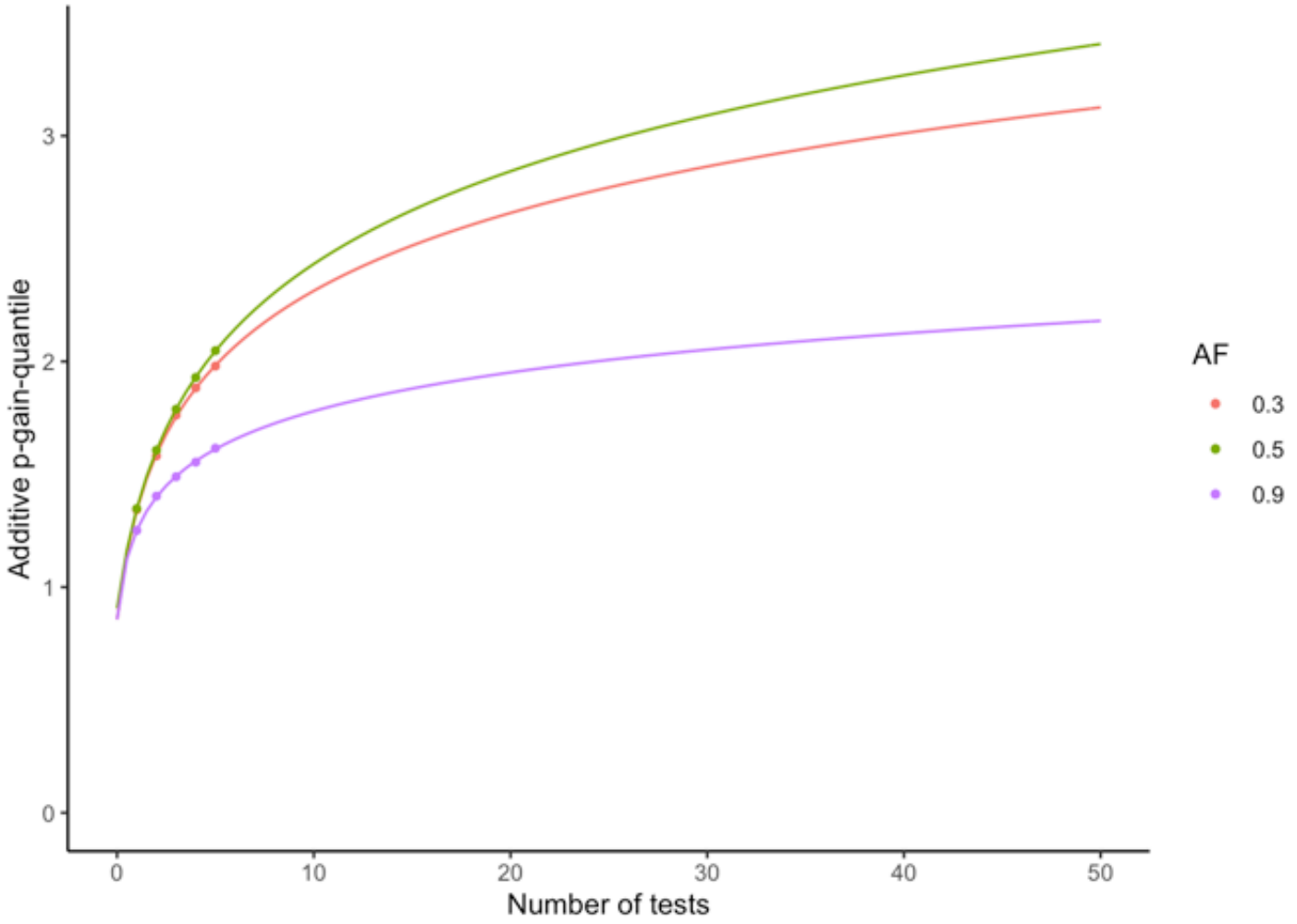
```
list_fits <- p_gain_quantile_fit(pgain_quantile=quantile, test_number=50L, print_pd
f=FALSE)
```

```
## [1] "Fitted function for recessive p-gain-quantiles of allele frequency 0.3"
## Nonlinear regression model
##   model: pgain_quantile[[index_rec]][, i] ~ log(a + b * number_tests,     base =
d)
##    data: parent.frame()
##      a      b      d
## 1.0399 0.1959 1.0482
##  residual sum-of-squares: 0.1359
##
## Number of iterations to convergence: 7
## Achieved convergence tolerance: 1.49e-08
## [1] "recessive p-gain-threshold for 50 tests, allele frequency 0.3"
##        a
## 50.65234
## [1] "Fitted function for recessive p-gain-quantiles of allele frequency 0.5"
## Nonlinear regression model
##   model: pgain_quantile[[index_rec]][, i] ~ log(a + b * number_tests,     base =
d)
##    data: parent.frame()
##      a      b      d
## 1.1912 0.3449 1.1398
##  residual sum-of-squares: 6.094e-05
##
## Number of iterations to convergence: 7
## Achieved convergence tolerance: 1.49e-08
## [1] "recessive p-gain-threshold for 50 tests, allele frequency 0.5"
##        a
## 22.27177
## [1] "Fitted function for recessive p-gain-quantiles of allele frequency 0.7"
## Nonlinear regression model
##   model: pgain_quantile[[index_rec]][, i] ~ log(a + b * number_tests,     base =
d)
##    data: parent.frame()
##     a     b     d
## 1.752 1.549 1.739
##  residual sum-of-squares: 0.003028
##
## Number of iterations to convergence: 14
## Achieved convergence tolerance: 1.49e-08
## [1] "recessive p-gain-threshold for 50 tests, allele frequency 0.7"
##        a
## 7.900926
## [1] "Fitted function for recessive p-gain-quantiles of allele frequency 0.9"
## Nonlinear regression model
##   model: pgain_quantile[[index_rec]][, i] ~ log(a + b * number_tests,     base =
d)
```

```
##    data: parent.frame()
##       a       b       d
##   7.567  23.989  11.037
##  residual sum-of-squares: 0.0001227
##
## Number of iterations to convergence: 22
## Achieved convergence tolerance: 1.49e-08
## [1] "recessive p-gain-threshold for 50 tests, allele frequency 0.9"
##        a
## 2.955134
```



```
## [1] "Fitted function for additive p-gain-quantiles of allele frequency 0.3"
## Nonlinear regression model
##   model: pgain_quantile[[index_add]][, j] ~ log(a + b * number_tests,     base =
d)
##    data: parent.frame()
##     a     b     d
## 5.174 7.957 6.816
##  residual sum-of-squares: 0.000113
##
## Number of iterations to convergence: 17
## Achieved convergence tolerance: 1.49e-08
## [1] "additive p-gain-threshold for 50 tests, allele frequency 0.3"
##        a
## 3.125718
## [1] "Fitted function for additive p-gain-quantiles of allele frequency 0.5"
## Nonlinear regression model
##   model: pgain_quantile[[index_add]][, j] ~ log(a + b * number_tests,     base =
d)
##    data: parent.frame()
##     a     b     d
## 4.164 4.184 4.827
##  residual sum-of-squares: 5.406e-06
##
## Number of iterations to convergence: 12
## Achieved convergence tolerance: 1.49e-08
## [1] "additive p-gain-threshold for 50 tests, allele frequency 0.5"
##        a
## 3.406811
## [1] "Fitted function for additive p-gain-quantiles of allele frequency 0.9"
## Nonlinear regression model
##   model: pgain_quantile[[index_add]][, j] ~ log(a + b * number_tests,     base =
d)
##    data: parent.frame()
##      a      b      d
##  30.00 114.03  52.93
##  residual sum-of-squares: 6.112e-05
##
## Number of iterations to convergence: 38
## Achieved convergence tolerance: 1.49e-08
## [1] "additive p-gain-threshold for 50 tests, allele frequency 0.9"
##        a
## 2.180313
```



By means of `start_vec` the starting estimates for the log-linear-fit for different p-gain types and allele frequencies can be changed.

In a GWAS setting the full estimation of critical values can be done as follows: A1) Run the GWAS with an additive, recessive and dominant coding and select index hits with p<1.67e-8 (=0.05/ 3 models / 1 million independent tests). A2) Run pgainsim assuming no genetic association adjusting for the same 3 million tests and cover the observed allele frequencies or span a grid of allele frequencies.

B1) the same as A1) Run the GWAS with an additive, recessive and dominant coding and select index hits with p<1.67e-8 (=0.05/ 3 models / 1 million independent tests). B2) Run pgainsim assuming the specific genetic effects you want to reject using the arguments `effect_size` (numeric value), `sim_MOI` (simulated genetic association: none, add, rec, dom) and `trait_variance` (trait variance within a genotype) in the function `p_gain_simulation()` . E.g. p-value_recessive is the lowest p-value, then choose `sim_MOI` ="add" and take the effect estimate of the additive fit as `effect_size` and the average across genotypes of within genotype trait variance as `trait_variance` and simulate pgains under this model. The number of tests one should correct for reduces in this case to the number of significant index SNPs.

Scenario A is a simplification that potentially allows for a computationally much more efficient setup (if you have many hits). Scenario B is the stringently theoretically correct approach. In practise with a metabolome-wide GWAS (1400 traits) we found out that scenario A was for most allele frequencies more conservative than scenario B.

Scenario A is outlined in above code. Scenario B is described below.

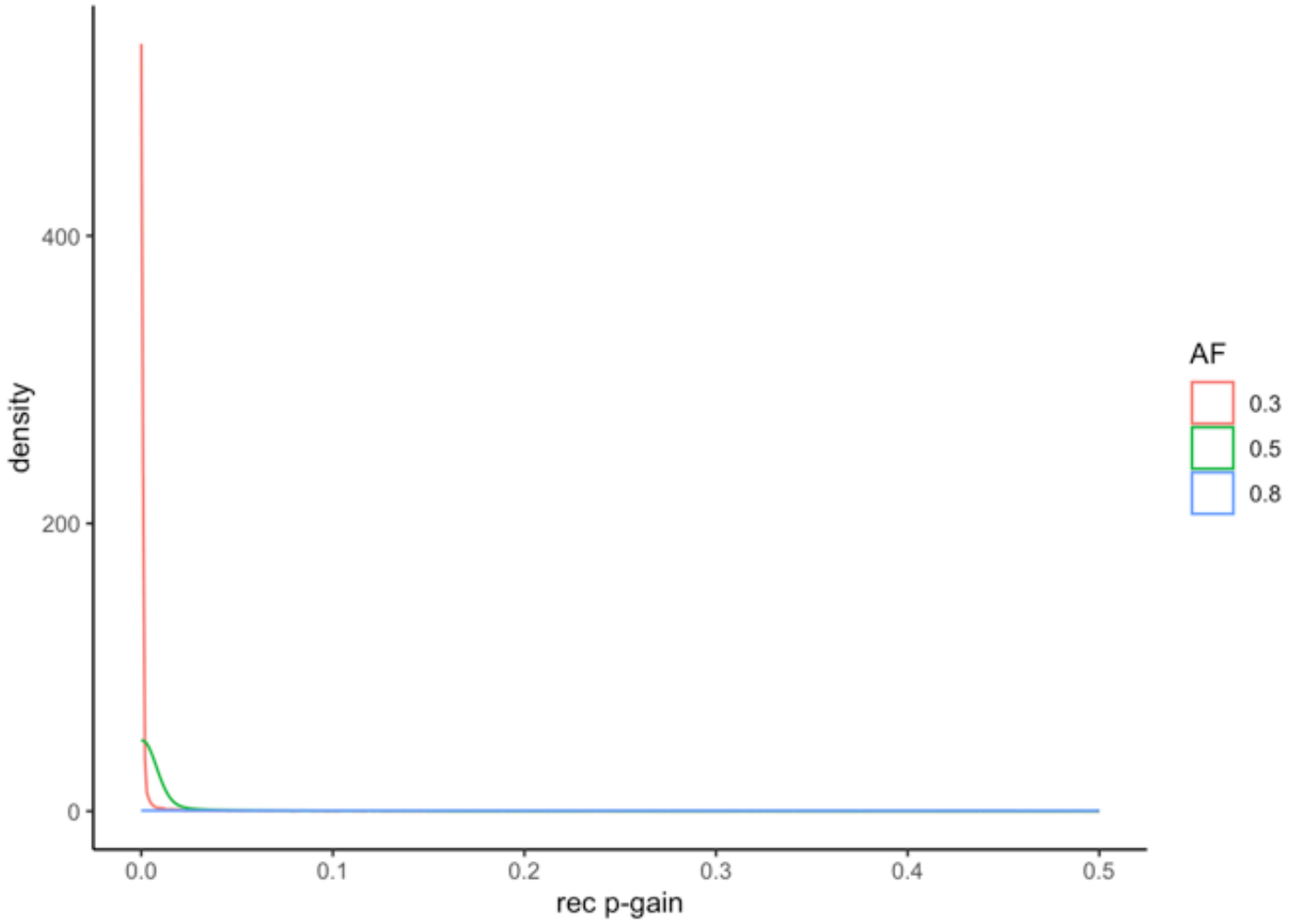
```
sim_dat <- p_gain_simulation(pgain_types="rec", AFs=c(0.3,0.5,0.8), effect_size=0.3
, trait_variance=0.8, sim_MOI="add", n=10000L, cores=1L)
```

```
## [1] "Generating 10000 random draws."
```

```
p_gain_density_plot(pgain_type="rec", sim_data=sim_dat, xlim=c(0,0.5), adjust=50, p
rint_pdf=FALSE)
```

```
## Warning: Removed 2820 rows containing non-finite values (`stat_density()`).
```

```
## Warning: Removed 1 rows containing missing values (`geom_vline()`).
```

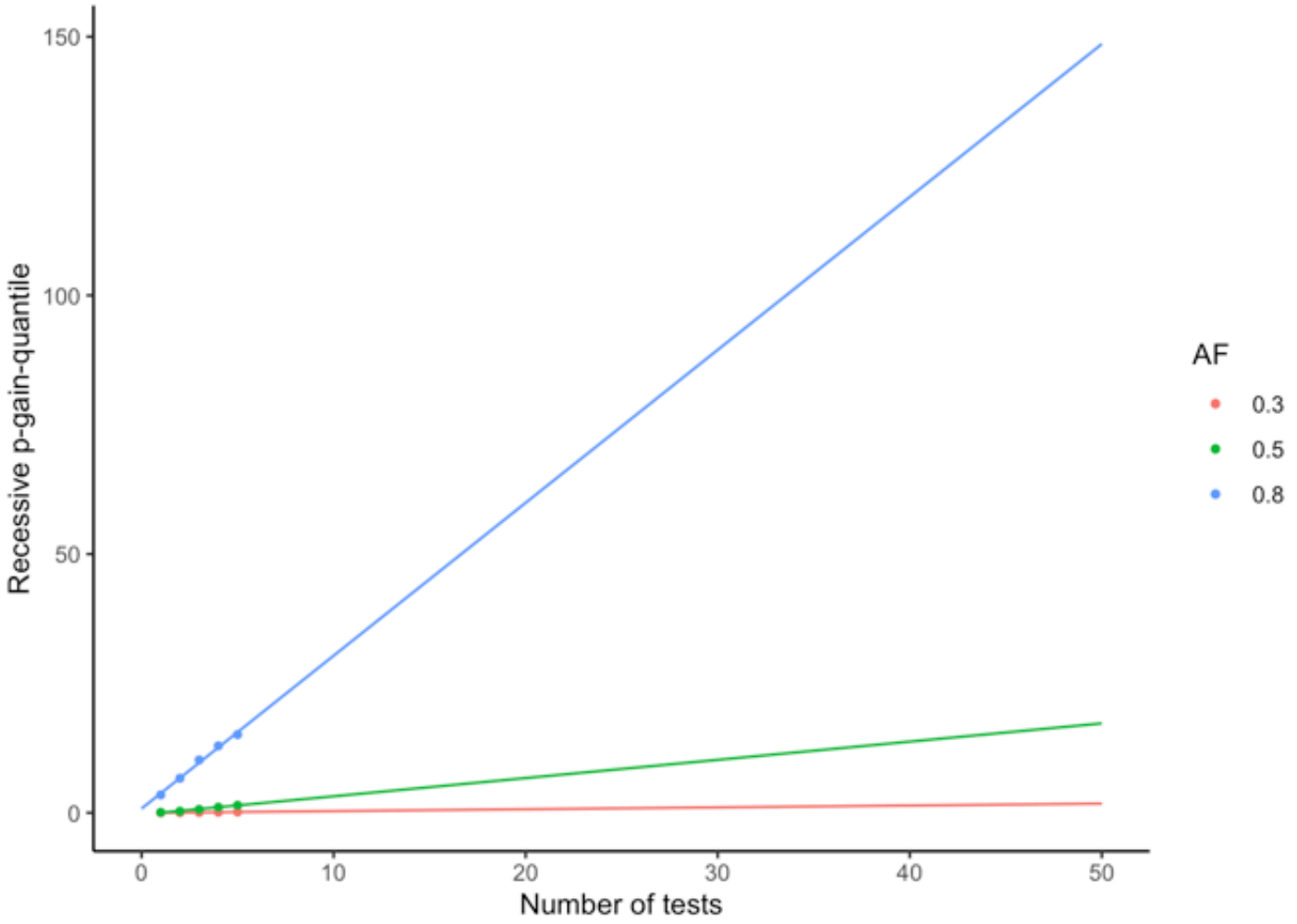



```
quantile <- p_gain_quantiles(n_tests=5L, sim_data=sim_dat)
vec <- list(data.frame("0.3"=c(a=1,b=0.00001,d=1.001),"0.5"=c(a=1.1,b=0.000001,d=1.
0000001), "0.8"=c(a=1,b=0.000001,d=1.00000001)))
list_fits <- p_gain_quantile_fit(pgain_quantile=quantile, test_number=50L, start_ve
c=vec, print_pdf=FALSE)
```

```
## [1] "Fitted function for recessive p-gain-quantiles of allele frequency 0.3"
## Nonlinear regression model
##   model: pgain_quantile[[index_rec]][, i] ~ log(a + b * number_tests,     base =
d)
##    data: parent.frame()
##         a         b         d
## 1.000e+00 2.945e-05 1.001e+00
##  residual sum-of-squares: 0.0001333
##
## Number of iterations to convergence: 9
## Achieved convergence tolerance: 1.49e-08
## [1] "recessive p-gain-threshold for 50 tests, allele frequency 0.3"
##        a
## 1.766523
## [1] "Fitted function for recessive p-gain-quantiles of allele frequency 0.5"
## Nonlinear regression model
##   model: pgain_quantile[[index_rec]][, i] ~ log(a + b * number_tests,     base =
d)
##    data: parent.frame()
##         a         b         d
## 1.000e+00 4.108e-06 1.000e+00
##  residual sum-of-squares: 0.01198
##
## Number of iterations to convergence: 25
## Achieved convergence tolerance: 1.49e-08
## [1] "recessive p-gain-threshold for 50 tests, allele frequency 0.5"
##       a
## 17.2768
## [1] "Fitted function for recessive p-gain-quantiles of allele frequency 0.8"
## Nonlinear regression model
##   model: pgain_quantile[[index_rec]][, i] ~ log(a + b * number_tests,     base =
d)
##    data: parent.frame()
##         a         b         d
## 1.000e+00 4.774e-07 1.000e+00
##  residual sum-of-squares: 0.691
##
## Number of iterations to convergence: 10
## Achieved convergence tolerance: 1.49e-08
## [1] "recessive p-gain-threshold for 50 tests, allele frequency 0.8"
##        a
## 148.5819
```

```
## Warning: Removed 2 rows containing missing values (`geom_line()`).
```

```
## Warning: Removed 2 rows containing missing values (`geom_line()`).
```



For details on the options of all functions please see `?pgainsim` and the corresponding application note by Scherer et al. 2021 (under review).

Session Info

```
sessionInfo()
```

```
## R version 4.1.2 (2021-11-01)
## Platform: aarch64-apple-darwin20 (64-bit)
## Running under: macOS 13.1
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.1-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.1-arm64/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
## [1] pgainsim_1.1.0   reshape2_1.4.4   ggplot2_3.4.0    minpack.lm_1.2-2
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_1.1.1  xfun_0.29         bslib_0.3.1       remotes_2.4.2
##  [5] purrr_0.3.4       colorspace_2.0-3  vctrs_0.5.1       generics_0.1.1
##  [9] testthat_3.1.1    usethis_2.1.5     htmltools_0.5.2   yaml_2.2.1
## [13] utf8_1.2.2        rlang_1.0.6       pkgbuild_1.3.1    jquerylib_0.1.4
## [17] pillar_1.8.1      glue_1.6.2        withr_2.5.0       DBI_1.1.2
## [21] sessioninfo_1.2.2 plyr_1.8.8        lifecycle_1.0.3   stringr_1.5.0
## [25] munsell_0.5.0     gtable_0.3.1      devtools_2.4.3    memoise_2.0.1
## [29] evaluate_0.14     labeling_0.4.2    knitr_1.37        callr_3.7.0
## [33] fastmap_1.1.0     ps_1.6.0          fansi_1.0.3       highr_0.9
## [37] Rcpp_1.0.9        scales_1.2.1      cachem_1.0.6      desc_1.4.0
## [41] pkgload_1.2.4     jsonlite_1.7.2    farver_2.1.1      fs_1.5.2
## [45] digest_0.6.29     stringi_1.7.12    processx_3.5.2    dplyr_1.0.7
## [49] rprojroot_2.0.2   grid_4.1.2        cli_3.6.0         tools_4.1.2
## [53] magrittr_2.0.3    sass_0.4.0        tibble_3.1.8      crayon_1.5.1
## [57] pkgconfig_2.0.3   ellipsis_0.3.2    prettyunits_1.1.1 assertthat_0.2.1
## [61] rmarkdown_2.11    R6_2.5.1          compiler_4.1.2
```

```
warnings()
```