F14C

Andrew M. Dolman

21 February 2018

To run this script you will need the following packages

```
library(ggplot2)
library(dplyr)
library(tidyr)
library(rstan)
library(egg)
```

Alkanoic data

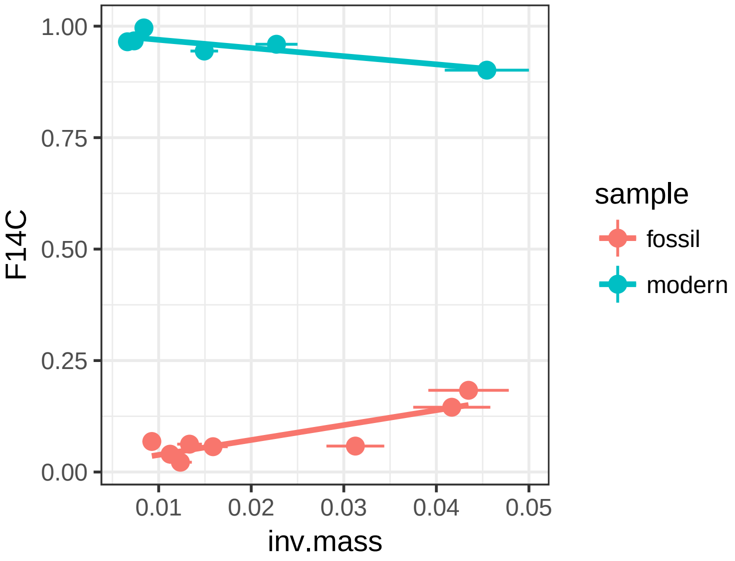
Load the data

```
alkanoic <- read.csv("alkanoic-data.csv")
```

Plot data to check it loaded correctly

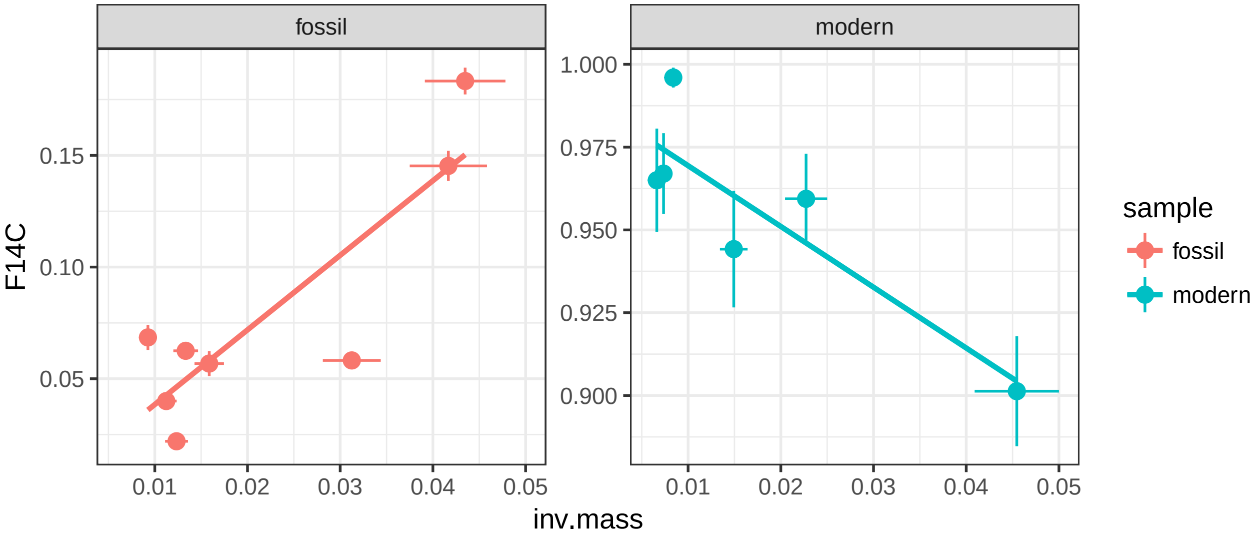
```
PlotData <- function(data){
  p <- ggplot(data=data, aes(x = inv.mass, y = F14C, colour = sample)) +
    geom_pointrange(aes(ymax = F14C + 2*F14C.err, ymin = F14C - 2*F14C.err)) +
    geom_segment(aes(xend = inv.mass + 2*inv.mass.err,
                     x = inv.mass - 2*inv.mass.err,
                     y = F14C, yend = F14C))+
  geom_smooth(method = "lm", se = F) +
  theme_bw()
  p
}
```

```
p.alk <- PlotData(alkanoic)
p.alk
```



For the fossil sample, the reported uncertainties for F14C are too small to account for the deviations from the regression line. An additional error term will be required.

```
p.alk + facet_wrap(~sample, scales = "free_y")
```



Fit Bayesian regression models

Bayesian modelling will allow an error model for both response and predictor variables. Posterior samples of regression parameters will allow for easy numerical estimation of distribution of intersection of the two regression lines.

The data and parameter values for a Stan model needs to be in a named list

```
# Function to arrange data for Stan model
GetStanData <- function(df,
                        mu_int_modern = 0, sigma_int_modern = 10,
                        mu_int_fossil = 0, sigma_int_fossil = 10){
  foss <- dplyr::filter(df, sample == "fossil")
  mod <- dplyr::filter(df, sample == "modern")
  stan.dat <- list(
    x_meas_modern = mod$inv.mass,
    x_meas_fossil = foss$inv.mass,

    x_err_modern = mod$inv.mass.err,
    x_err_fossil = foss$inv.mass.err,

    y_err_modern = mod$F14C.err,
    y_err_fossil = foss$F14C.err,

    y_meas_modern = mod$F14C,
    y_meas_fossil = foss$F14C,

    N_modern = length(mod$inv.mass),
    N_fossil = length(foss$inv.mass),

    mu_int_modern = mu_int_modern,
    sigma_int_modern = sigma_int_modern,

    mu_int_fossil = mu_int_fossil,
    sigma_int_fossil = sigma_int_fossil
  )
  return(stan.dat)
  }
```

The Stan model is defined in a separate text file.

```
mod_code <- readLines("F14C-calibration_y_err_b.Stan")
```

```
# create the data object
stan.dat.alk <- GetStanData(alkanoic,
                            mu_int_modern = 0.9705, sigma_int_modern = 0.0036,
                            mu_int_fossil = 0.0003, sigma_int_fossil = 0.0002)

# Set seed so that results are replicable (same set of random numbers used)
set.seed(20180328)

# compile and fit the model
stan.fit.alk <- rstan::stan(model_code = mod_code,
                            data = stan.dat.alk, iter = 5000,
                            cores = 3, chains = 3, init = "random")
```

Summary of the fitted model. The Rhat values should be close to 1.

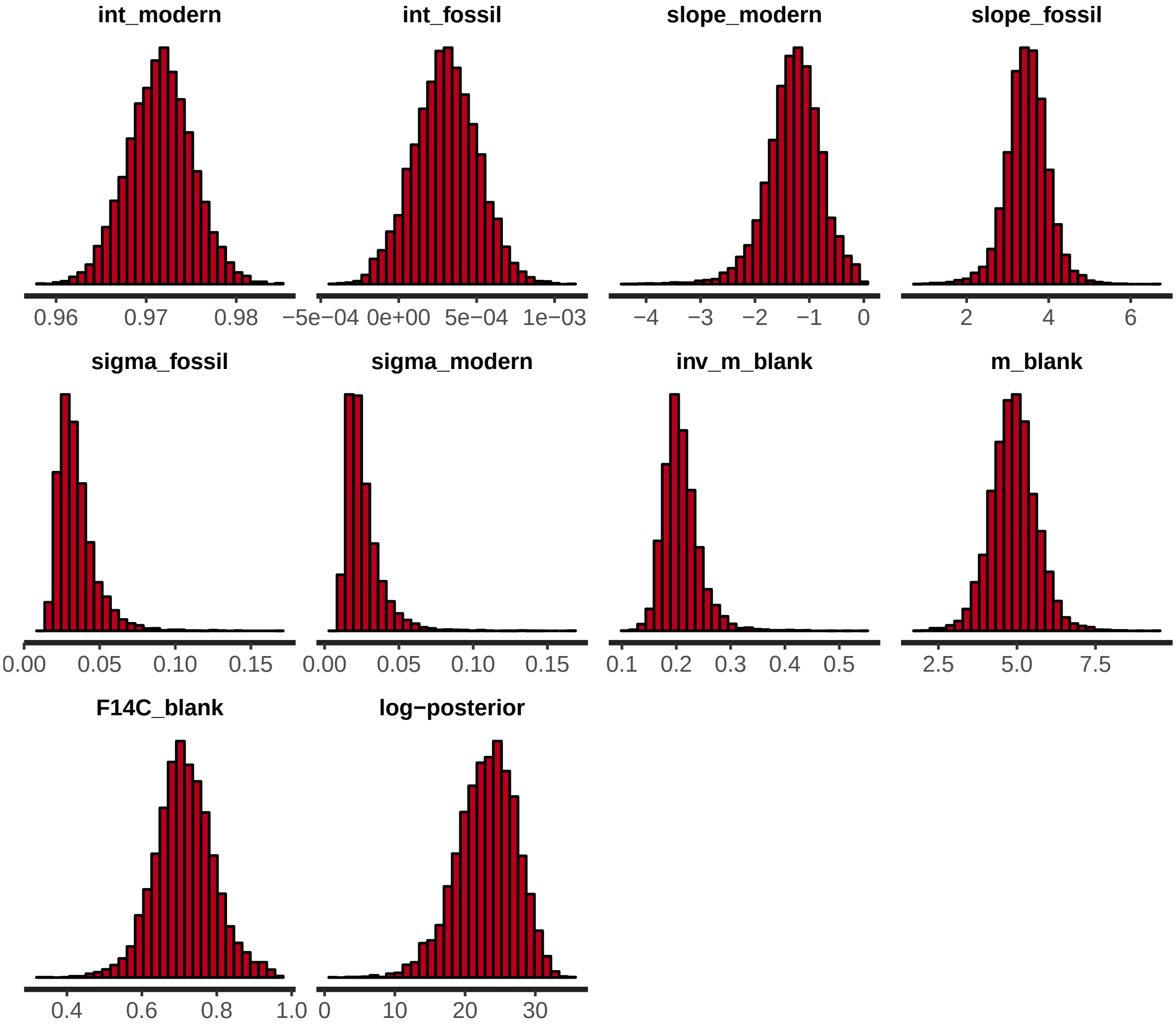
```
print(stan.fit.alk)
```

```
## Inference for Stan model: 08ec63bd3d2e76a3421b07cb5f557244.
## 3 chains, each with iter=5000; warmup=2500; thin=1;
## post-warmup draws per chain=2500, total post-warmup draws=7500.
##
##                  mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## x_modern[1]      0.01    0.00 0.00  0.01  0.01  0.01  0.01  0.01  7500    1
## x_modern[2]      0.01    0.00 0.00  0.01  0.01  0.01  0.01  0.01  7500    1
## x_modern[3]      0.01    0.00 0.00  0.01  0.01  0.01  0.01  0.01  7500    1
## x_modern[4]      0.01    0.00 0.00  0.01  0.01  0.01  0.02  0.02  7500    1
## x_modern[5]   0.02    0.00 0.00  0.02  0.02  0.02  0.02  0.03  7500    1
## x_modern[6]   0.05    0.00 0.00  0.04  0.04  0.05  0.05  0.05  7500    1
## x_fossil[1]   0.01    0.00 0.00  0.01  0.01  0.01  0.01  0.01  7500    1
## x_fossil[2]   0.02    0.00 0.00  0.01  0.02  0.02  0.02  0.02  7500    1
## x_fossil[3]   0.04    0.00 0.00  0.04  0.04  0.04  0.04  0.05  7500    1
## x_fossil[4]   0.01    0.00 0.00  0.01  0.01  0.01  0.01  0.01  7500    1
## x_fossil[5]   0.03    0.00 0.00  0.03  0.03  0.03  0.03  0.03  7500    1
## x_fossil[6]   0.04    0.00 0.00  0.04  0.04  0.04  0.05  0.05  7500    1
## x_fossil[7]   0.01    0.00 0.00  0.01  0.01  0.01  0.01  0.01  6970    1
## x_fossil[8]   0.01    0.00 0.00  0.01  0.01  0.01  0.01  0.01  7500    1
## y_modern[1]   0.96    0.00 0.01  0.95  0.96  0.96  0.97  0.98  7500    1
## y_modern[2]   0.97    0.00 0.01  0.96  0.96  0.97  0.97  0.98  7500    1
## y_modern[3]   1.00    0.00 0.00  0.99  0.99  1.00  1.00  1.00  7500    1
## y_modern[4]   0.95    0.00 0.01  0.93  0.94  0.95  0.95  0.96  7500    1
## y_modern[5]   0.96    0.00 0.01  0.95  0.95  0.96  0.96  0.97  7500    1
## y_modern[6]   0.90    0.00 0.01  0.89  0.90  0.90  0.91  0.92  7500    1
## y_fossil[1]   0.04    0.00 0.00  0.04  0.04  0.04  0.04  0.04  7500    1
## y_fossil[2]   0.06    0.00 0.00  0.05  0.05  0.06  0.06  0.06  7500    1
## y_fossil[3]   0.15    0.00 0.00  0.14  0.14  0.15  0.15  0.15  7500    1
## y_fossil[4]   0.02    0.00 0.00  0.02  0.02  0.02  0.02  0.02  7500    1
## y_fossil[5]   0.06    0.00 0.00  0.05  0.06  0.06  0.06  0.06  7500    1
## y_fossil[6]   0.18    0.00 0.00  0.18  0.18  0.18  0.18  0.19  7500    1
## y_fossil[7]   0.07    0.00 0.00  0.06  0.07  0.07  0.07  0.07  7500    1
## y_fossil[8]   0.06    0.00 0.00  0.06  0.06  0.06  0.06  0.07  7500    1
## int_modern    0.97    0.00 0.00  0.96  0.97  0.97  0.97  0.98  7500    1
## int_fossil    0.00    0.00 0.00  0.00  0.00  0.00  0.00  0.00  7500    1
## slope_modern -1.29    0.01 0.50 -2.36 -1.57 -1.26 -0.96 -0.35  4297    1
## slope_fossil  3.47    0.01 0.51  2.45  3.17  3.47  3.78  4.46  3585    1
## sigma_fossil  0.03    0.00 0.01  0.02  0.03  0.03  0.04  0.07  3926    1
## sigma_modern  0.03    0.00 0.01  0.01  0.02  0.02  0.03  0.06  4136    1
## inv_m_blank   0.21    0.00 0.03  0.16  0.19  0.20  0.23  0.29  3180    1
## m_blank       4.90    0.01 0.73  3.49  4.44  4.89  5.33  6.37  3979    1
## F14C_blank    0.71    0.00 0.08  0.56  0.66  0.71  0.76  0.88  3602    1
## lp__         23.03    0.08 4.22 13.91 20.32 23.34 26.03 30.35  2755    1
##
## Samples were drawn using NUTS(diag_e) at Sun Apr 22 16:29:33 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

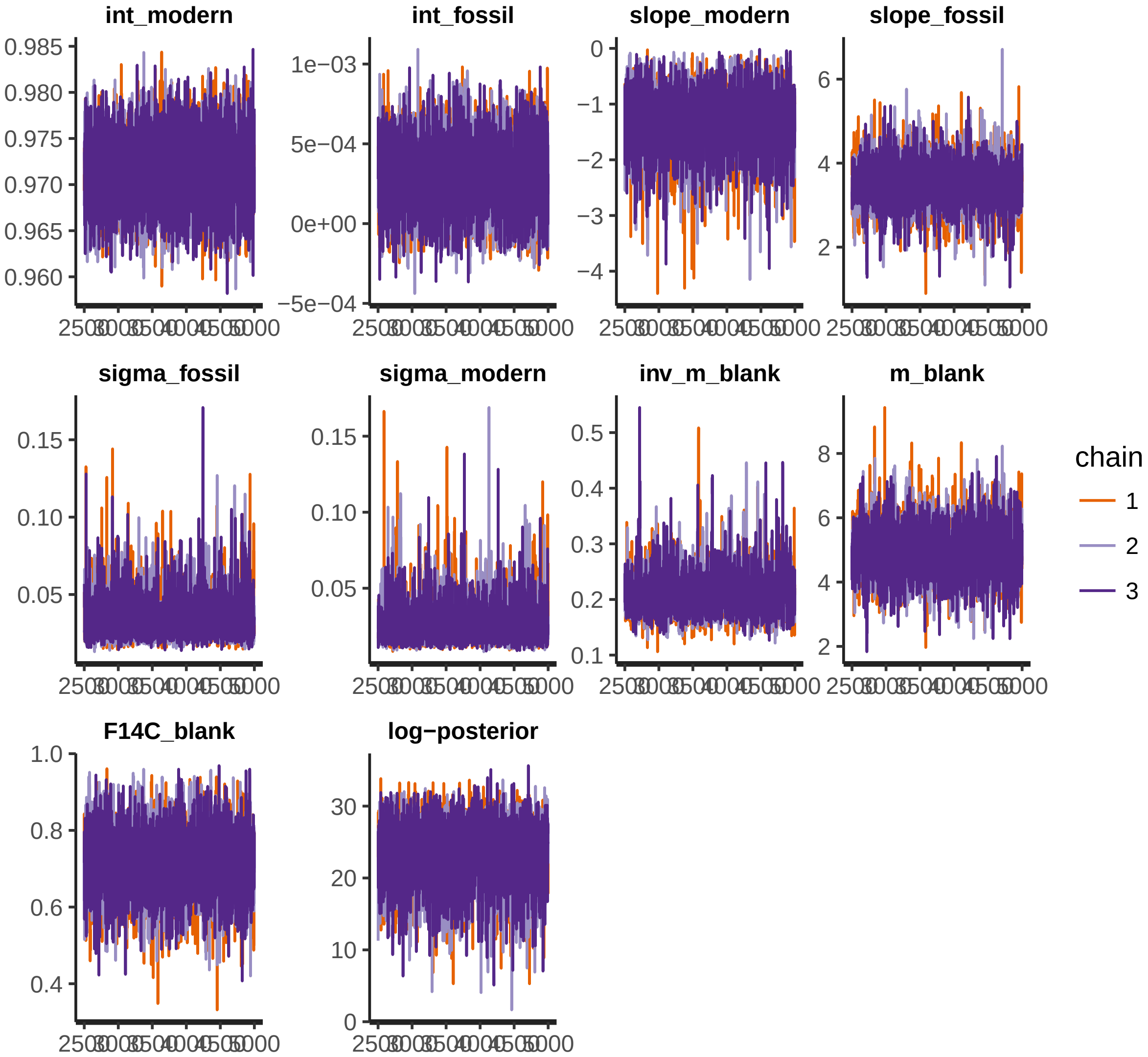
The rstan package includes default methods for plotting the fitted model, e.g., histograms of the estimated parameters and traceplots to check that the chains have converged.

```
rstan::stan_hist(stan.fit.alk,
                 pars = c("x_modern", "x_fossil", "y_modern", "y_fossil"),
                 include = FALSE)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```r
rstan::traceplot(stan.fit.alk,
                 pars = c("x_modern", "x_fossil", "y_modern", "y_fossil"),
                 include = FALSE, inc_warmup = FALSE)
```



Extract posterior distribution

The output from a Bayesian model is the "posterior distribution" which consists of a matrix of parameters estimates. Each row gives a set of estimates from a single iteration of the sampler. There were 3 chains of 5000 iterations each, the first 2500 iterations of each chain are discarded as warm up. So we get 7500 iterations to work with.

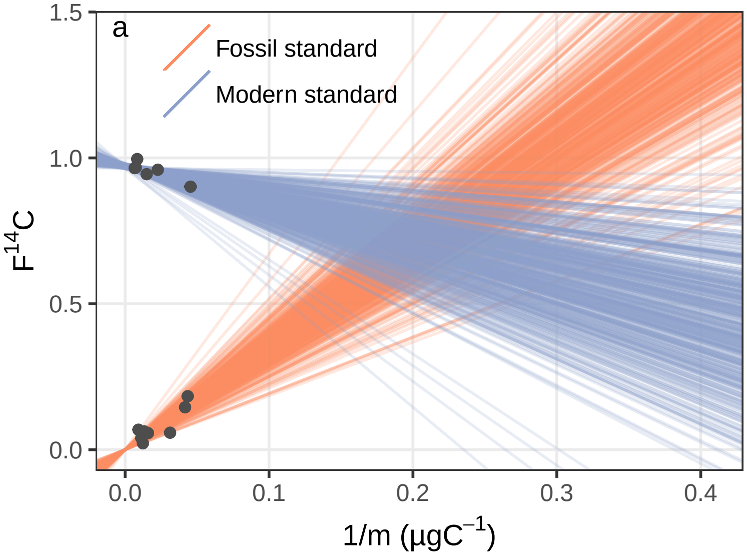
```
# Extract posterior distribution
post.dist.alk <- as_tibble(rstan::extract(stan.fit.alk,
                                          pars = c("int_modern", "int_fossil",
                                                   "slope_modern", "slope_fossil",
                                                   "inv_m_blank", "m_blank", "F14C_blank")))

post.dist.alk
```

```
## # A tibble: 7,500 x 7
##    int_modern int_fossil slope_modern slope_fossil inv_m_blank m_blank
##         <dbl>      <dbl>        <dbl>        <dbl>       <dbl>   <dbl>
##  1      0.972   0.000311        -1.51         3.15       0.208    4.80
##  2      0.974   0.000265       -1.13          2.45       0.272    3.68
##  3      0.970   0.000496       -2.09          2.89       0.195    5.14
##  4      0.972   0.000118       -1.33          3.29       0.210    4.76
##  5      0.974   0.000304       -0.774         3.48       0.229    4.37
##  6      0.977   0.000522       -1.23          3.55       0.204    4.90
##  7      0.973   0.000454       -1.39          3.46       0.200    4.99
##  8      0.972   0.000296       -1.41          4.09       0.177    5.66
##  9      0.973   0.000395       -0.955         3.52       0.217    4.61
## 10      0.971   0.000519       -1.35          3.64       0.194    5.15
## # ... with 7,490 more rows, and 1 more variable: F14C_blank <dbl>
```

Plot the estimated slopes as a visual check of the model. Plotting all 20000 estimates is too many so we plot a subsample.

```
PlotPosteriorSlopes <- function(data, posterior){
  # Use 2x the median value of the intersection as the axis limits
  max.x <- 2 * median(posterior$inv_m_blank)
  max.y <- 2 * median(posterior$F14C_blank)

  p <- ggplot(data = data,
                     aes(x = inv.mass, y = F14C, colour = sample)) +
  geom_abline(data = posterior,
              aes(intercept = int_fossil, slope = slope_fossil,
                  colour = "Fossil standard"), alpha = 0.2) +
  geom_abline(data = posterior,
              aes(intercept = int_modern, slope = slope_modern,
                  colour = "Modern standard"), alpha = 0.2) +
  geom_linerange(aes(ymax = F14C + 2*F14C.err, ymin = F14C - 2*F14C.err),
                  colour = "grey30", alpha = 1) +
  geom_segment(aes(xend = inv.mass + 2*inv.mass.err, x = inv.mass - 2*inv.mass.err,
                   y = F14C, yend = F14C),
                  colour = "grey30", alpha = 1) +
    geom_point(colour = "grey30", alpha = 1) +
  scale_x_continuous(expression("1/m (µgC"^-1*")"), limits = c(0, max.x)) +
  scale_y_continuous(expression("F"^14*"C"), limits = c(0, max.y)) +
    scale_colour_manual(values = c("Fossil standard" = "#fc8d62",
                                   "Modern standard" = "#8da0cb"))+
  guides(color = guide_legend(override.aes = list(alpha = 1))) +
    theme_bw() +
    theme(legend.position = c(0, 1), legend.justification = c(-0.2, 0.9),
          legend.title = element_blank(), legend.background = element_blank(),
          panel.grid.minor = element_blank())
  return(p)
}

PlotIntersections <- function(posterior){
  upr <- function(x){median(x) + mad(x)}
  lwr <- function(x){median(x) - mad(x)}

  p.w <- dplyr::summarise_all(posterior,
                              funs(median = median, sd = sd, mad = mad,
                                   lwr = lwr, upr = upr))

  p.ints <- ggplot(data = posterior) +
    geom_point(aes(x = m_blank, y = F14C_blank), alpha = 0.02, size = 1.5) +
    geom_linerange(data = p.w,
             aes(x = m_blank_median, y = F14C_blank_median,
                 ymax = F14C_blank_upr, ymin = F14C_blank_lwr),
             colour = "Red", size = 1, alpha = 0.8) +
    geom_errorbarh(data = p.w,
                   aes(x = m_blank_median, y = F14C_blank_median,
                       xmax = m_blank_upr, xmin = m_blank_lwr,
                       height = 0), colour = "Red", size = 1, alpha = 0.8) +
    scale_x_continuous(expression("m"[blank]*" (µgC)")) +
    scale_y_continuous(expression("F"^14*"C"[blank])) +
    theme_bw() +
    theme(panel.grid.minor = element_blank())
  p.ints
}
```

```
# Number of lines to plot
n.samples <- 500

post.dist.samp.alk <- post.dist.alk[sample(1:nrow(post.dist.alk), n.samples), ]

p.slps.alk <- PlotPosteriorSlopes(data = alkanoic, posterior = post.dist.samp.alk) +
  annotate("text", label = "a", x = -Inf, y = Inf, hjust = -1, vjust = 1.2)
p.slps.alk
```

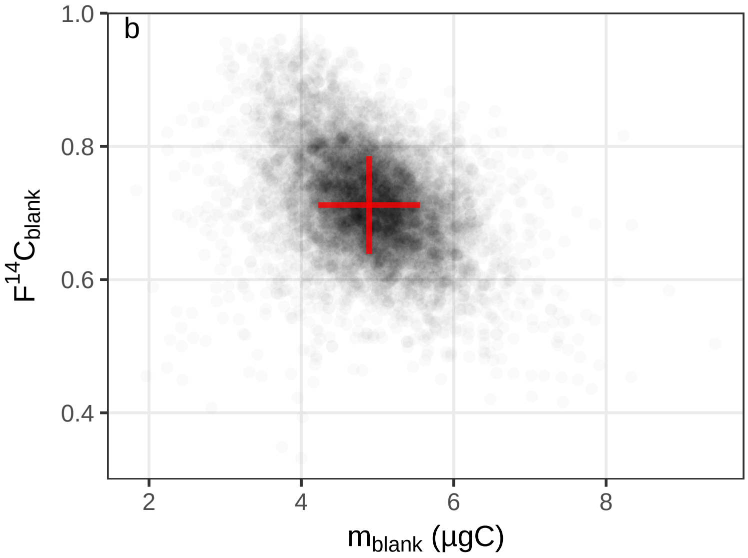


Plot all the intersections of the two regression lines. The intersections are the paired estimates of F14C_blank and inv_m_blank.

```
p.ints.alk <- PlotIntersections(post.dist.alk) +
  annotate("text", label = "b", x = -Inf, y = Inf, hjust = -1, vjust = 1.2)
```

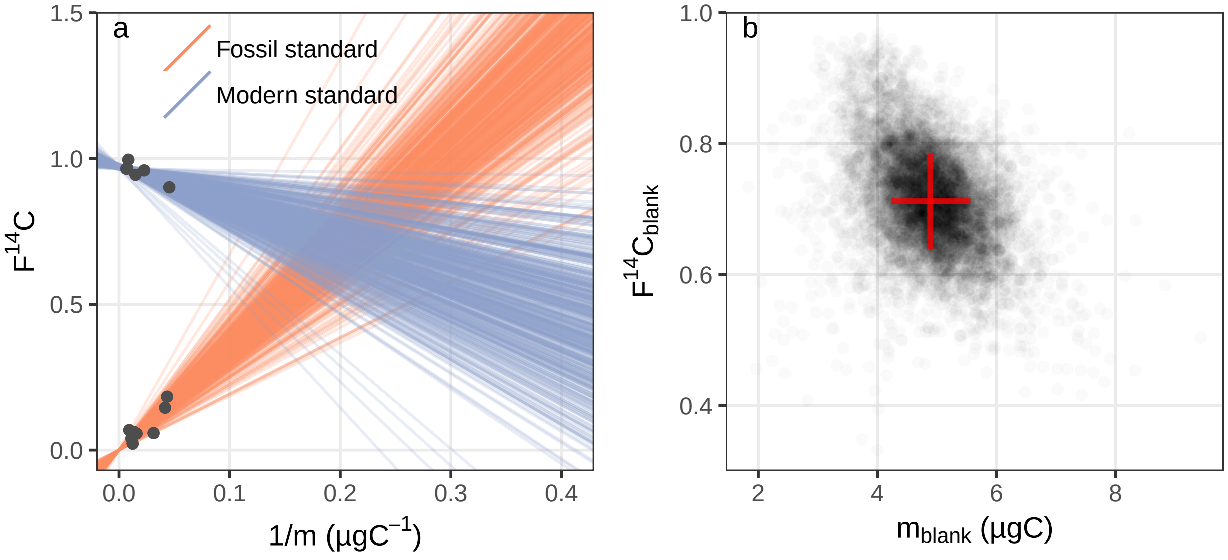
```
## Warning: Ignoring unknown aesthetics: y
```

```
p.ints.alk
```



A combined figure.

```
egg::ggarrange(p.slps.alk, p.ints.alk, ncol = 2)
```



Summary statistics

The median, 2.5% and 97.5% quantiles can be used as point estimates + CI of the values of inv_m_blank and F14C_blank. The median absolute deviation (MAD) might be better than the standard deviation as a measure of uncertainty for error propagation because the intersection is the ratio of the differences in slopes and intercepts, so its distribution has very long tails (is not Gaussian).

```
tab.alk <- post.dist.alk %>%
  gather(parameter, value) %>%
  group_by(parameter) %>%
  summarise_all(funs(mean = mean, sd = sd, median = median, MAD = mad,
                     lwr = quantile(., 0.025), upr = quantile(., 0.975))) %>%
  mutate("Standard compound" = "n-alkanoic acid") %>%
  select("Standard compound", everything())
```

```r
knitr::kable(tab.alk, digits = 3)
```

Standard compound	parameter	mean	sd	median	MAD	lwr	upr
n-alkanoic acid	F14C_blank	0.714	0.080	0.712	0.073	0.556	0.882
n-alkanoic acid	int_fossil	0.000	0.000	0.000	0.000	0.000	0.001
n-alkanoic acid	int_modern	0.972	0.004	0.972	0.004	0.964	0.979
n-alkanoic acid	inv_m_blank	0.209	0.034	0.205	0.028	0.157	0.287
n-alkanoic acid	m_blank	4.898	0.729	4.889	0.665	3.488	6.368
n-alkanoic acid	slope_fossil	3.472	0.506	3.473	0.450	2.446	4.464
n-alkanoic acid	slope_modern	-1.285	0.501	-1.263	0.450	-2.359	-0.353

Lignin data

Here we repeat the above analysis for the Lignin data.

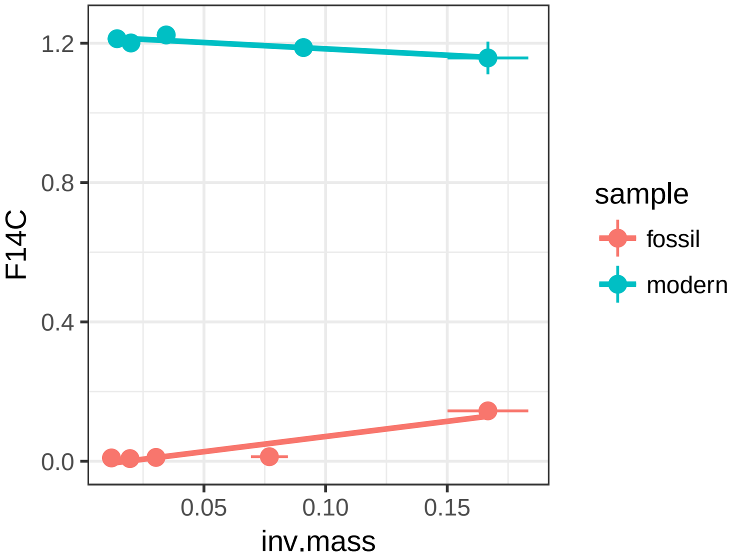
Load data

```r
lignin <- read.csv("lignin-data.csv", stringsAsFactors = FALSE)
head(lignin)
```

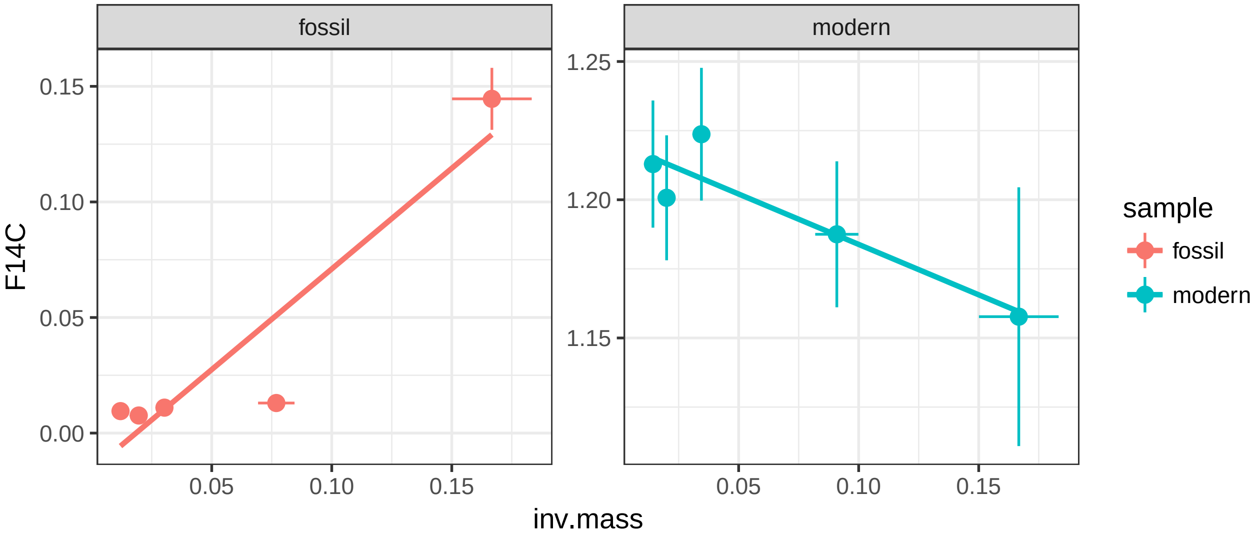
```
##   mass inv.mass inv.mass.err   F14C F14C.err sample substance
## 1   83   0.0120       0.0006 0.0095   0.0013 fossil    lignin
## 2   51   0.0196       0.0010 0.0076   0.0012 fossil    lignin
## 3   33   0.0303       0.0015 0.0110   0.0013 fossil    lignin
## 4   13   0.0769       0.0038 0.0130   0.0016 fossil    lignin
## 5    6   0.1667       0.0083 0.1446   0.0067 fossil    lignin
## 6   70   0.0143       0.0007 1.2129   0.0115 modern    lignin
```

Plot data to check it loaded correctly

```r
p.lig <- PlotData(lignin)
p.lig
```



```
p.lig + facet_wrap(~sample, scales = "free_y")
```



Fit Stan model

The fitted model for the Alkanoic data can be updated using new data. This avoids the model having to be compiled again.

```
stan.dat.lig <- GetStanData(lignin, mu_int_modern = 0, sigma_int_modern = 10,
                            mu_int_fossil = 0.0002, sigma_int_fossil = 0.0004)

# Fit model to lignin data set
stan.fit.lig <- rstan::stan(fit = stan.fit.alk, iter = 5000,
                        data = stan.dat.lig, cores = 3, chains = 3)
```

Check fitted model

Summary

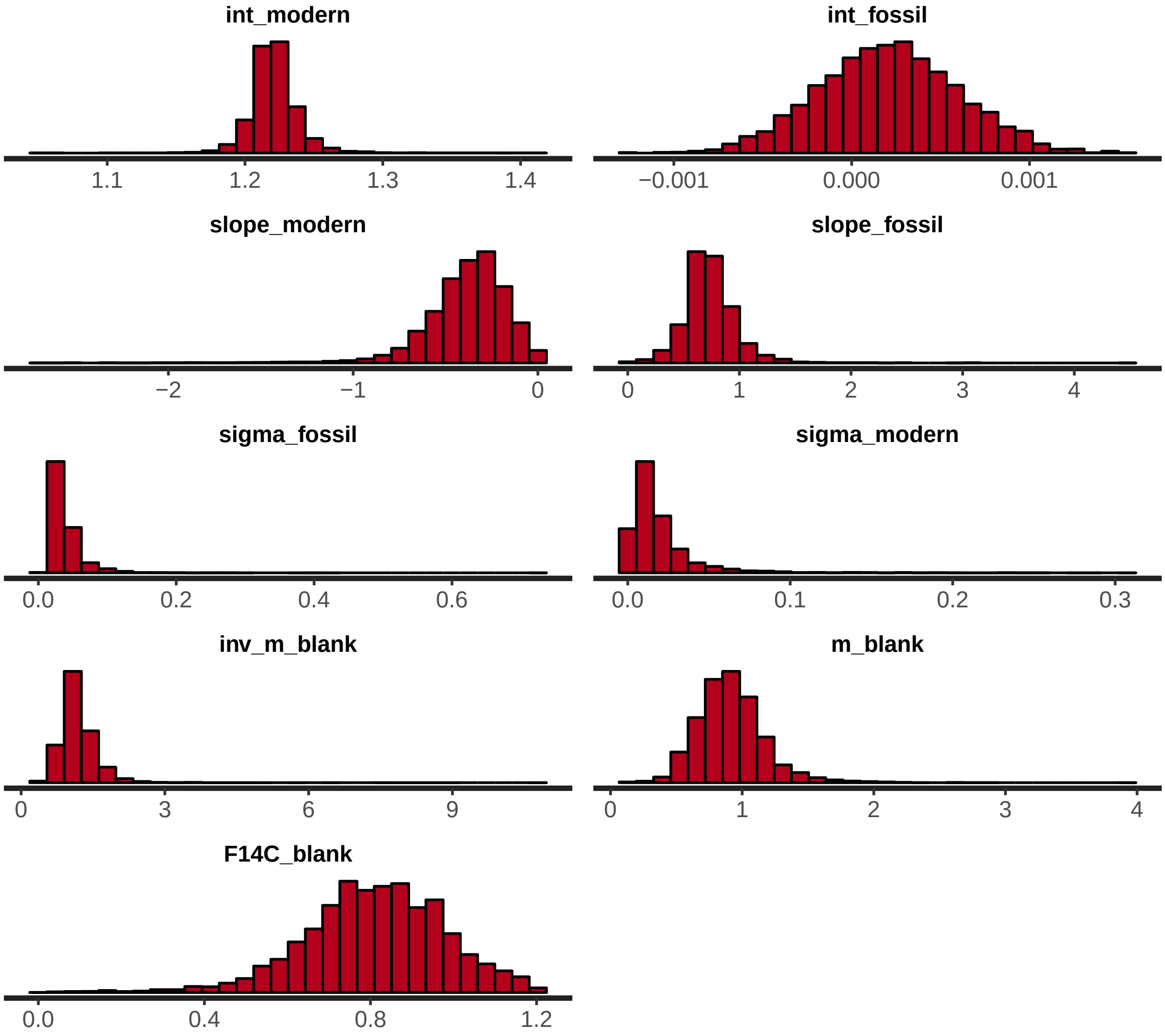
```
stan.fit.lig
```

```
## Inference for Stan model: 08ec63bd3d2e76a3421b07cb5f557244.
## 3 chains, each with iter=5000; warmup=2500; thin=1; 
## post-warmup draws per chain=2500, total post-warmup draws=7500.
## 
##                mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## x_modern[1]   0.01    0.00 0.00  0.01  0.01  0.01  0.01  0.02  1993 1.00
## x_modern[2]   0.02    0.00 0.00  0.02  0.02  0.02  0.02  0.02   328 1.01
## x_modern[3]   0.03    0.00 0.00  0.03  0.03  0.03  0.04  0.04   546 1.01
## x_modern[4]   0.09    0.00 0.00  0.08  0.09  0.09  0.09  0.10  1249 1.00
## x_modern[5]   0.17    0.00 0.01  0.15  0.16  0.17  0.17  0.18   616 1.00
## x_fossil[1]   0.01    0.00 0.00  0.01  0.01  0.01  0.01  0.01   452 1.01
## x_fossil[2]   0.02    0.00 0.00  0.02  0.02  0.02  0.02  0.02   227 1.01
## x_fossil[3]   0.03    0.00 0.00  0.03  0.03  0.03  0.03  0.03   440 1.01
## x_fossil[4]   0.08    0.00 0.00  0.07  0.07  0.08  0.08  0.08   830 1.01
## x_fossil[5]   0.17    0.00 0.01  0.15  0.16  0.17  0.17  0.18   474 1.01
## y_modern[1]   1.21    0.00 0.01  1.19  1.21  1.21  1.22  1.23  2646 1.00
## y_modern[2]   1.21    0.00 0.01  1.19  1.20  1.21  1.21  1.22  1402 1.00
## y_modern[3]   1.22    0.00 0.01  1.20  1.21  1.22  1.22  1.24   894 1.00
## y_modern[4]   1.19    0.00 0.01  1.17  1.18  1.19  1.19  1.21  1575 1.00
## y_modern[5]   1.16    0.00 0.02  1.12  1.15  1.16  1.17  1.20   943 1.00
## y_fossil[1]   0.01    0.00 0.00  0.01  0.01  0.01  0.01  0.01  7500 1.00
## y_fossil[2]   0.01    0.00 0.00  0.01  0.01  0.01  0.01  0.01  7500 1.00
## y_fossil[3]   0.01    0.00 0.00  0.01  0.01  0.01  0.01  0.01   308 1.01
## y_fossil[4]   0.01    0.00 0.00  0.01  0.01  0.01  0.01  0.02  1970 1.00
## y_fossil[5]   0.14    0.00 0.01  0.13  0.14  0.14  0.15  0.16   329 1.01
## int_modern    1.22    0.00 0.02  1.19  1.21  1.22  1.23  1.26  2116 1.00
## int_fossil    0.00    0.00 0.00  0.00  0.00  0.00  0.00  0.00  1031 1.00
## slope_modern -0.39    0.01 0.23 -0.89 -0.49 -0.36 -0.24 -0.05   780 1.00
## slope_fossil  0.73    0.01 0.24  0.30  0.60  0.72  0.85  1.23   452 1.01
## sigma_fossil  0.04    0.00 0.03  0.02  0.02  0.03  0.04  0.10   940 1.00
## sigma_modern  0.02    0.00 0.02  0.00  0.01  0.01  0.02  0.07   790 1.00
## inv_m_blank   1.19    0.01 0.44  0.66  0.95  1.12  1.34  2.02   929 1.00
## m_blank       0.91    0.01 0.26  0.49  0.74  0.89  1.05  1.50   684 1.00
## F14C_blank    0.81    0.01 0.17  0.44  0.71  0.82  0.93  1.13   461 1.00
## lp__         15.24    0.21 4.66  5.18 12.34 15.56 18.62 23.31   515 1.00
## 
## Samples were drawn using NUTS(diag_e) at Sun Apr 22 16:29:52 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at 
## convergence, Rhat=1).
```

Histograms

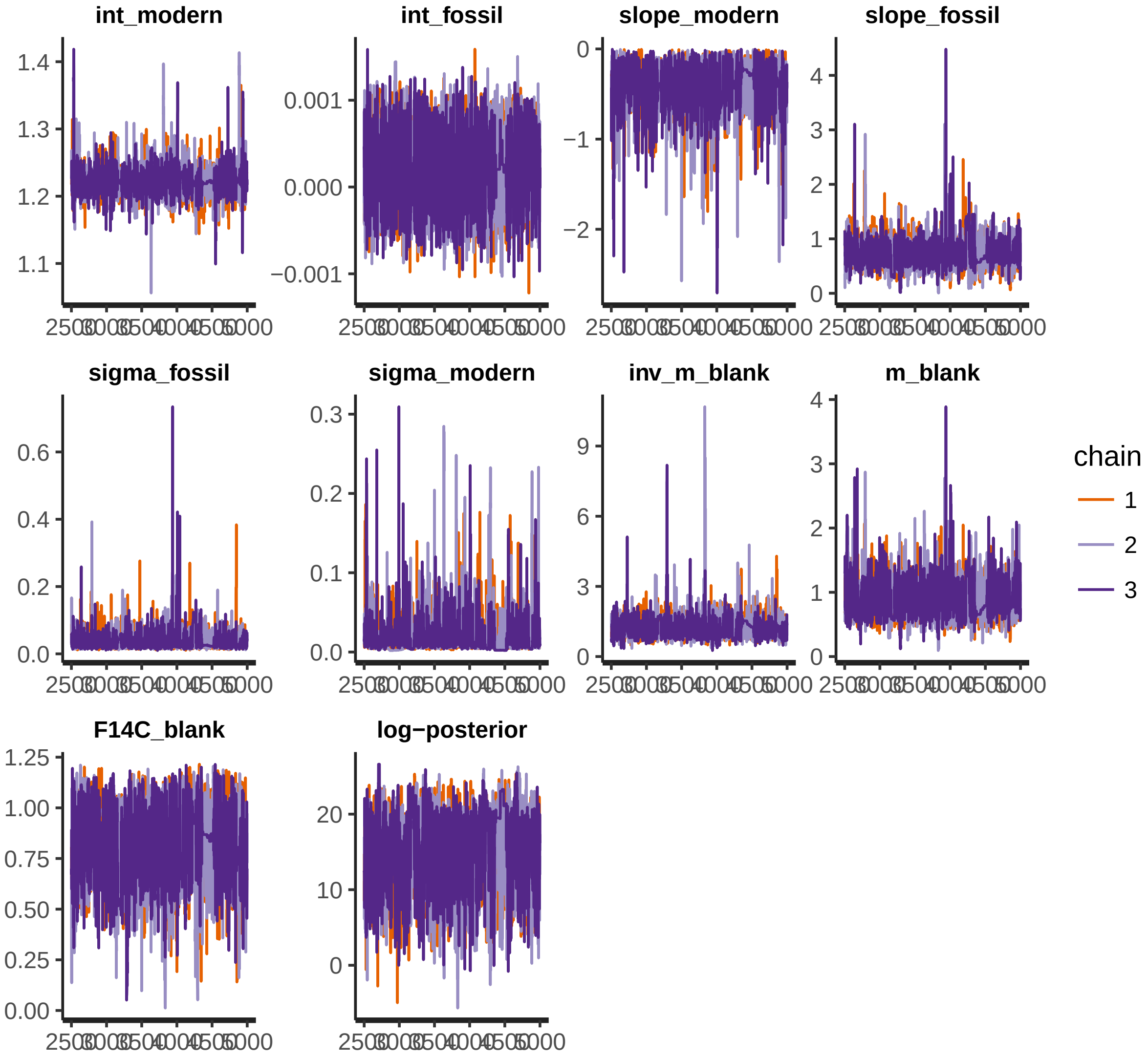
```
p <- rstan::stan_hist(stan.fit.lig,
                      pars = c("x_modern", "x_fossil", "y_modern",
                               "y_fossil", "log-posterior"),
                      include = FALSE, ncol = 2)
p
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Traceplots

```
rstan::traceplot(stan.fit.lig, pars = c("x_modern", "x_fossil", "y_modern", "y_fossil"),
                 include = FALSE, inc_warmup = FALSE)
```



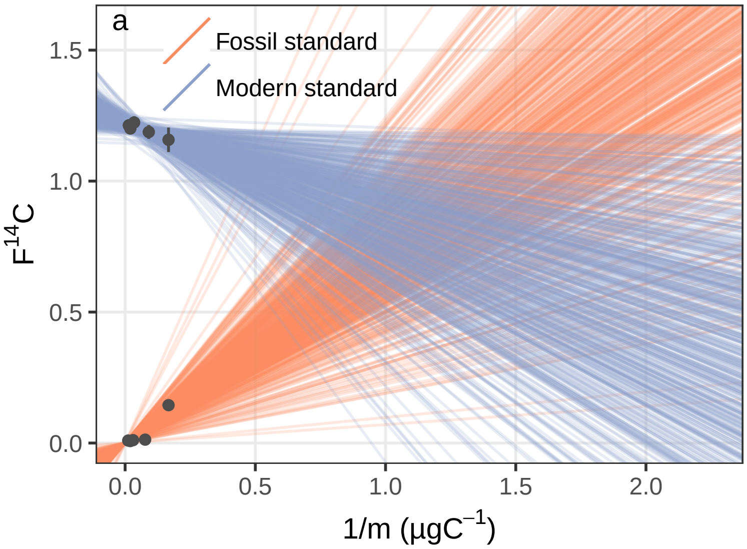
Extact posterior distribution

```r
# Extract posterior distribution
post.dist.lig <- as_tibble(rstan::extract(stan.fit.lig,
                                          pars = c("int_modern", "int_fossil",
                                                   "slope_modern", "slope_fossil",
                                                   "inv_m_blank", "m_blank",
                                                   "F14C_blank")))
```

Plot the estimated slopes as a visual check of the model. Plotting all 20000 estimates is too many, plot a subsample.

```r
# Number of lines to plot
n.samples <- 500

post.dist.samp.lig <- post.dist.lig[sample(1:nrow(stan.fit.lig), n.samples), ]

p.slps.lig <- PlotPosteriorSlopes(data = lignin, posterior = post.dist.samp.lig) +
  annotate("text", label = "a", x = -Inf, y = Inf, hjust = -1, vjust = 1.2)
```

```
p.slps.lig
```

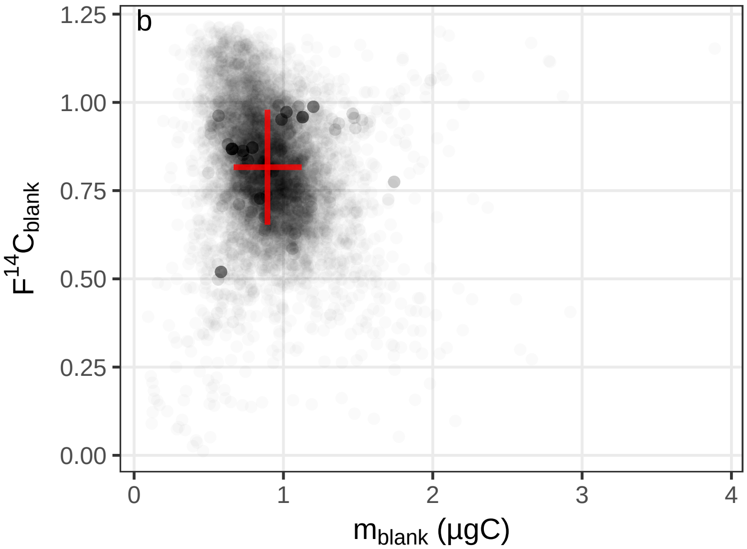


Plot all the intersections of the two regression lines. The intersections are the paired estimates of F14C_blank and inv_m_blank.

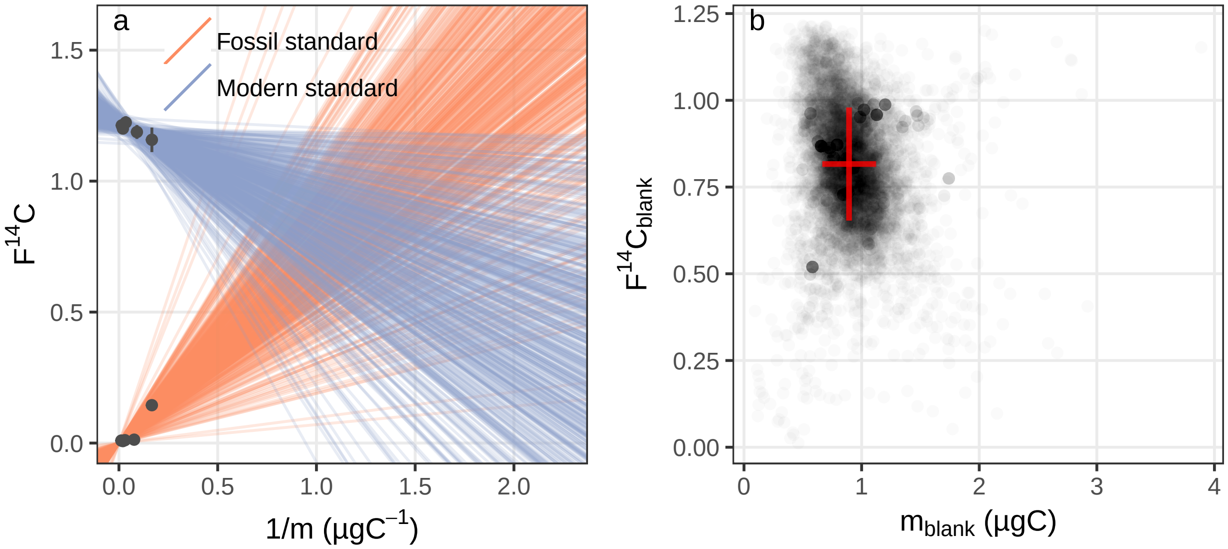
```
p.ints.lig <- PlotIntersections(post.dist.lig) +
  annotate("text", label = "b", x = -Inf, y = Inf, hjust = -1, vjust = 1.2)
```

```
## Warning: Ignoring unknown aesthetics: y
```

```
p.ints.lig
```



```
egg::ggarrange(p.slps.lig, p.ints.lig, ncol = 2)
```



Summary statistics

The median, 2.5% and 97.5% quantiles can be used as point estimates + CI of the values of inv_m_blank and F14C_blank. The median absolute deviation (MAD) might be better than the standard deviation as a measure of uncertainty for error propagation because the intersection is the ratio of the differences in slopes and intercepts, so its distribution has very long tails (is not Gaussian).

```
tab.lig <- post.dist.lig %>%
  gather(parameter, value) %>%
  group_by(parameter) %>%
  summarise_all(funs(mean = mean, sd = sd, median = median, MAD = mad,
                     lwr = quantile(., 0.025), upr = quantile(., 0.975)))%>%
  mutate("Standard compound" = "Lignin phenols") %>%
  select("Standard compound", everything())

knitr::kable(tab.lig, digits = 3)
```

Standard compound	parameter	mean	sd	median	MAD	lwr	upr
Lignin phenols	F14C_blank	0.810	0.171	0.816	0.163	0.444	1.127
Lignin phenols	int_fossil	0.000	0.000	0.000	0.000	-0.001	0.001
Lignin phenols	int_modern	1.221	0.019	1.220	0.013	1.189	1.260
Lignin phenols	inv_m_blank	1.190	0.437	1.120	0.280	0.664	2.024
Lignin phenols	m_blank	0.915	0.262	0.893	0.226	0.494	1.505
Lignin phenols	slope_fossil	0.733	0.240	0.716	0.186	0.302	1.234
Lignin phenols	slope_modern	-0.386	0.230	-0.355	0.184	-0.892	-0.050

Combined results

```
bind_rows(tab.alk, tab.lig) %>%
  filter(parameter %in% c("m_blank", "F14C_blank")) %>%
  select(-lwr, -upr) %>%
  knitr::kable(., digits = 3)
```

Standard compound	parameter	mean	sd	median	MAD
n-alkanoic acid	F14C_blank	0.714	0.080	0.712	0.073
n-alkanoic acid	m_blank	4.898	0.729	4.889	0.665
Lignin phenols	F14C_blank	0.810	0.171	0.816	0.163
Lignin phenols	m_blank	0.915	0.262	0.893	0.226