Bayesian design and analysis of external pilot trials for complex interventions: Appendix

D. T. Wilson

11 October 2018

Illustrative example - TIGA-CUB

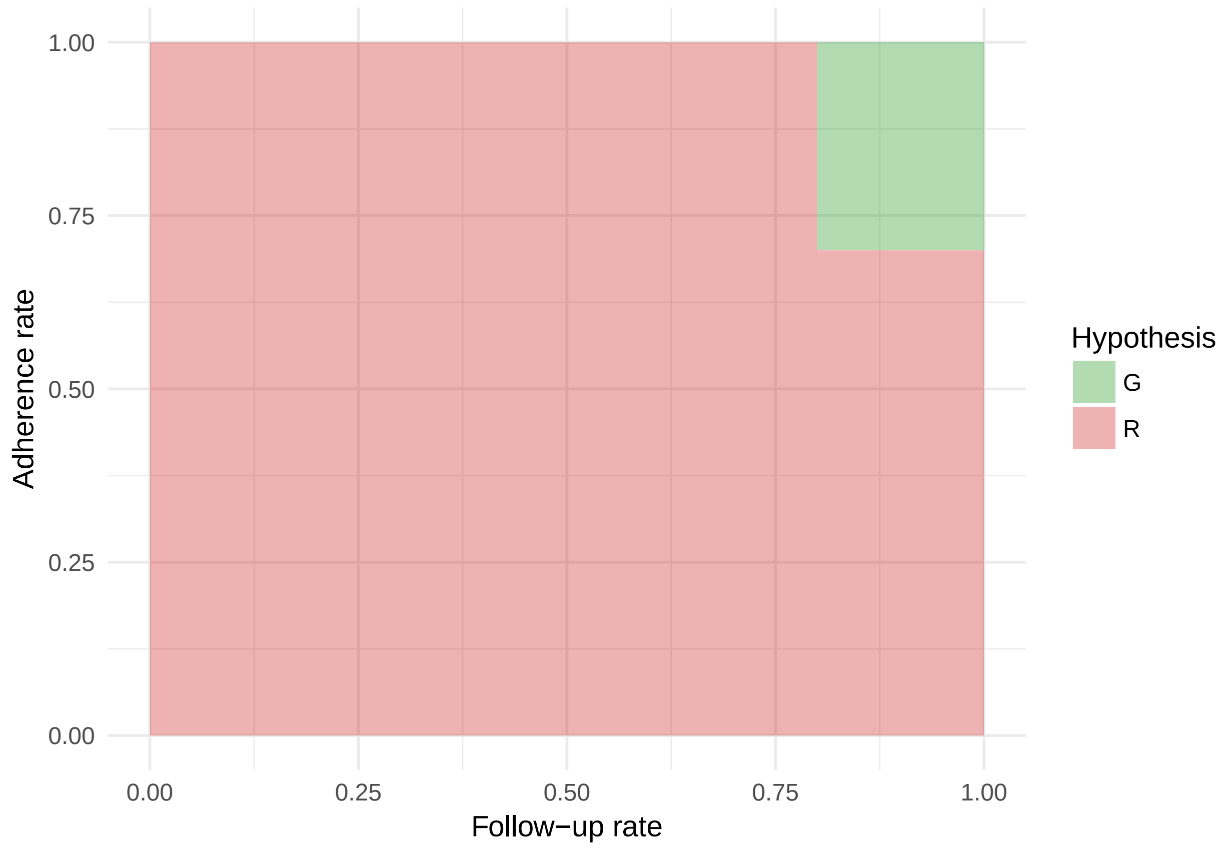
Hypotheses

Recall we are interested in the overall follow-up rate, p_f, and in the adherence rate in the intervention arm, p_a. The hypotheses described in the paper are:

```r
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.4.4
```

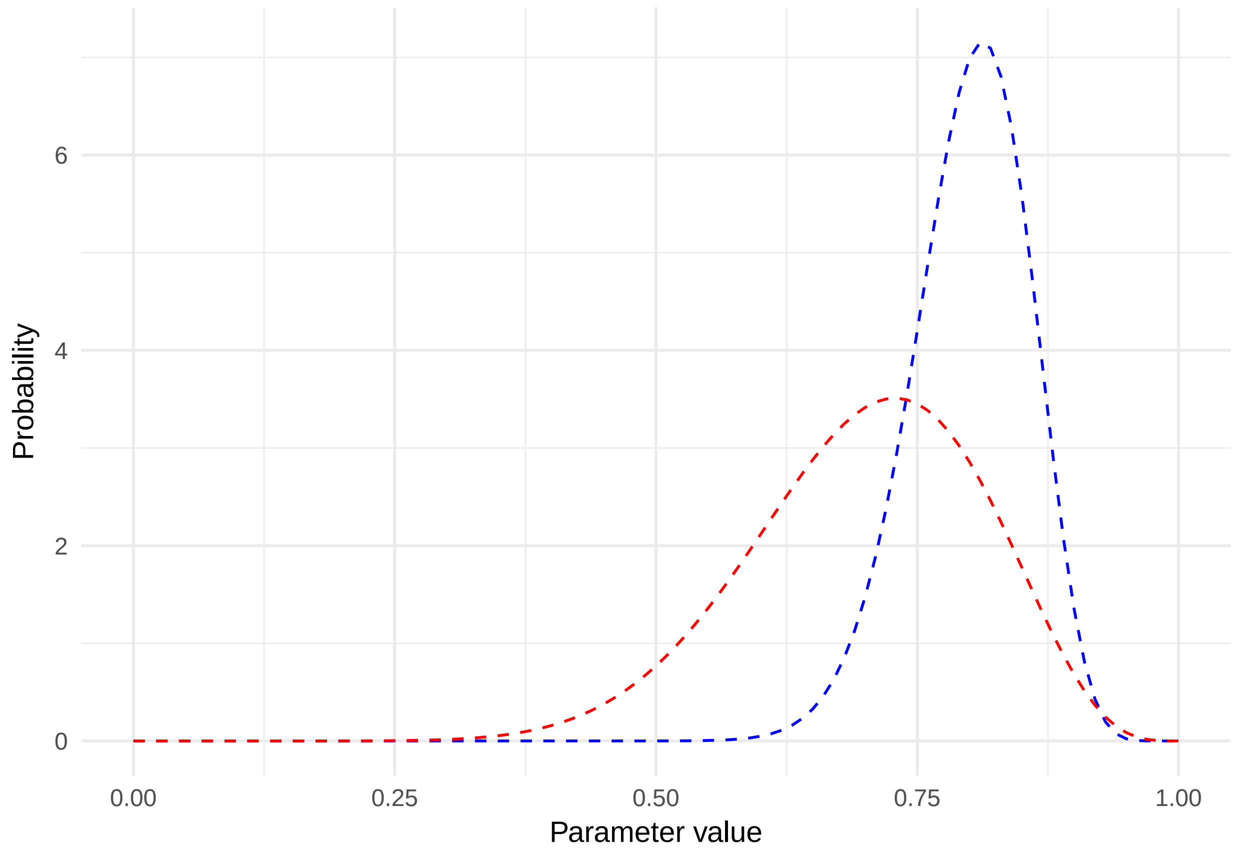
```r
# Threshold values for follow-up and adherence
f_c <- 0.8
a_c <- 0.7

ids <- factor(c("R", "G"))

values <- data.frame(
  id = ids,
  value = c("R", "G")
)

positions <- data.frame(
  id = c(rep("R", 6), rep("G", 4)),
  x = c(0, 0, f_c, f_c, 1, 1,    f_c, f_c, 1, 1),
  y = c(0, 1, 1, a_c, a_c, 0,    a_c, 1, 1, a_c)
)

datapoly <- merge(values, positions, by = c("id"))

ggplot(datapoly, aes(x = x, y = y)) +
  geom_polygon(aes(fill = value, group = id), alpha=0.3) +
  scale_fill_manual(name="Hypothesis",
                    labels=c("G", "R"),
                    values=c("green4","red3")) +
  xlab("Follow-up rate") + ylab("Adherence rate") +
  theme_minimal()
```



Priors

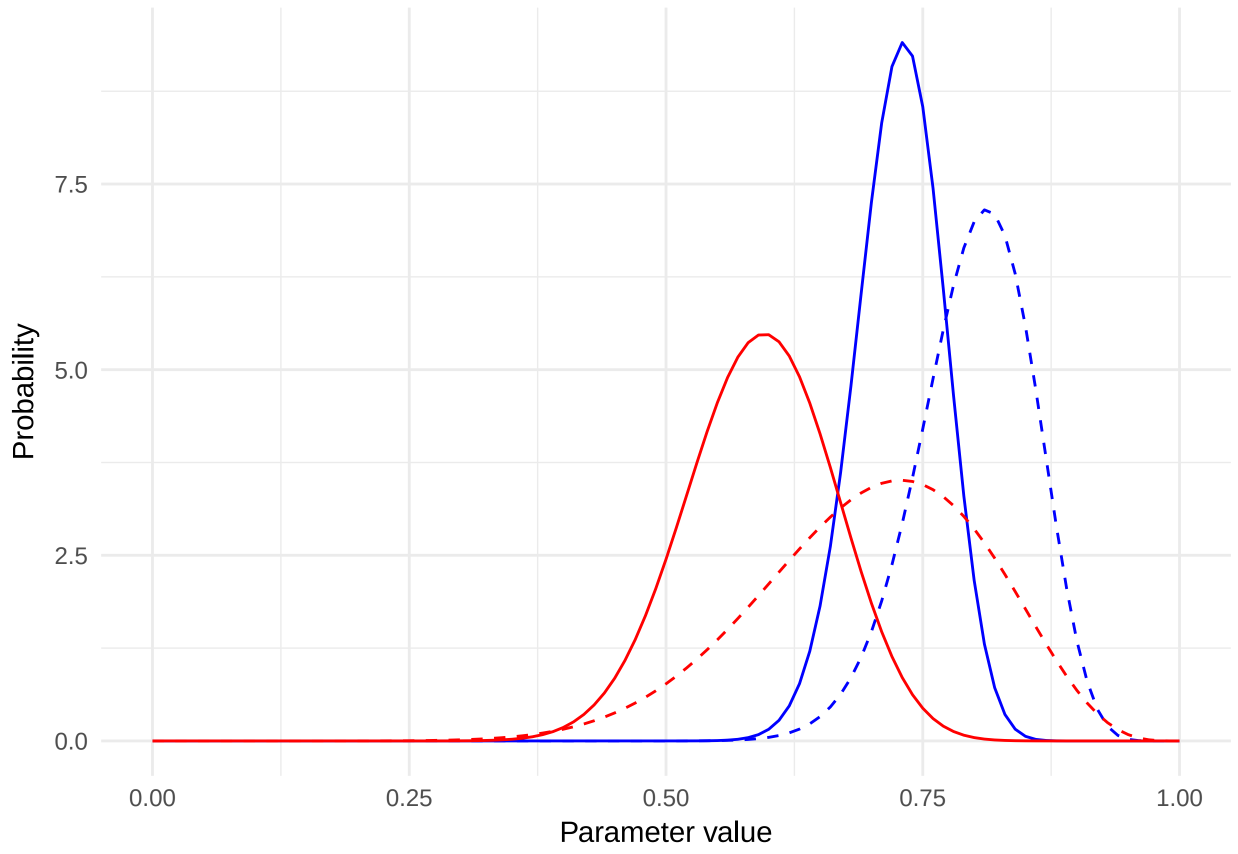
Adherence and follow-up are both measured at the individual level. We assume that both the number of follow-ups and the number of adherers follow binomial distributions with parameters p_f and p_a respectively. We use the following Beta design priors:

```
# Priors, parameterised by mean and effective sample size
f_mean <- 0.8; f_ss <- 50
f_a <- f_mean*f_ss; f_b <- f_ss - f_a

a_mean <- 0.7; a_ss <- 16
a_a <- a_mean*a_ss; a_b <- a_ss - a_a

# Plot priors and posteriors
ggplot(data.frame(x = seq(0,1,0.01)), aes(x)) +
  stat_function(fun = dbeta, args=list(shape1=f_a, shape2=f_b), linetype=2, colour="blue") +
  stat_function(fun = dbeta, args=list(shape1=a_a, shape2=a_b), linetype=2, colour="red") +
  theme_minimal() + xlab("Parameter value") + ylab("Probability")
```



Since the Beta distribution is conjugate for the binomial likelihood, we can easily obtain the posterior distribution given some data. For example:

```
# Data
n <- 30; f <- 40; a <- 16

f_a2 <- f_a+f; f_b2 <- f_b + (2*n - f)
a_a2 <- a_a + a; a_b2 <- a_b + (n - a)

# Plot priors and posteriors
ggplot(data.frame(x = seq(0,1,0.01)), aes(x)) +
  stat_function(fun = dbeta, args=list(shape1=f_a, shape2=f_b), linetype=2, colour="blue") +
  stat_function(fun = dbeta, args=list(shape1=f_a2, shape2=f_b2), linetype=1, colour="blue") +
  stat_function(fun = dbeta, args=list(shape1=a_a, shape2=a_b), linetype=2, colour="red") +
  stat_function(fun = dbeta, args=list(shape1=a_a2, shape2=a_b2), linetype=1, colour="red") +
  theme_minimal() + xlab("Parameter value") + ylab("Probability")
```



Operating characteristics

Recall that the operating characteristics of interest take the form

$$
\begin{aligned}
Pr[g, \phi \in \Phi_R] &= \int_{\Phi_R} Pr[g \mid \phi] p(\phi) d\phi \\
&= \int_{\Phi_R} \left(\sum_{x_f=0}^{n} \left[\sum_{x_a=0}^{n/2} \mathbb{I}(p_G < c_1 \mid x_f, x_a, n) p(x_a \mid \phi) \right] p(x_f \mid \phi) \right) p(\phi) d\phi,
\end{aligned}
$$

We implement two functions for estimating the operating characteristics from a design with sample size n and loss function parameter $c_1 = c$. The first, `get_ocs()`, uses a Monte Carlo approach to approximate the integration over the parameter space while calculating the summations exactly.

```
get_prob_g <- function(phi, df, n, c)
{
  p_f <- phi[1]; p_a <- phi[2]
  prob_f <- dbinom(df$f, size=2*n, prob=p_f)
  prob_a <- dbinom(df$a, size=n, prob=p_a)

  sum((df$post_f*df$post_a > c)*prob_f*prob_a)
}

get_ocs <- function(c, n, N, f_c, a_c, des_pri, an_pri=des_pri)
{
  # des/an _pri = c(f_a, f_b, a_a, a_b)
  df <- expand.grid(f=0:(2*n), a=0:n)
  df$post_f <- sapply(df$f, function(x, f_a, f_b, n) 1-pbeta(f_c, f_a+x, f_b+2*n-x),
                   f_a=an_pri[1], f_b=an_pri[2], n=n)
  df$post_a <- sapply(df$a, function(x, a_a, a_b, n) 1-pbeta(a_c, a_a+x, a_b+n-x),
                   a_a=an_pri[3], a_b=an_pri[4], n=n)
  phis <- data.frame(p_f = rbeta(N, des_pri[1], des_pri[2]), p_a = rbeta(N, des_pri[3], des_pri[4]))
  phis$g <- apply(phis, 1, get_prob_g, df=df, n=n, c=c)
  phis$H <- ifelse(phis$p_f > f_c & phis$p_a > a_c, "G", "R")

  # OC1, OC2
  c(sum(phis[phis$H == "R","g"])/N, sum(1-phis[phis$H == "G","g"])/N)
}

# For example,
get_ocs(c=0.5, n=30, N=10^4, f_c=0.8, a_c=0.7, des_pri=c(f_a, f_b, a_a, a_b), an_pri=rep(1,4))
```

```
## [1] 0.05709616 0.13753572
```

The second function is less precise for equal N but substantially faster`get_ocs2()`, using a Monte Carlo approximation for both the integration and the summations. It calls the c++ function `get_ocs_cpp()` to maximise speed.

```
// [[Rcpp::depends(BH)]]
// [[Rcpp::depends(RcppEigen)]]
// [[Rcpp::depends(RcppNumerical)]]

#include <Rcpp.h>
#include <RcppNumerical.h>
using namespace Numer;

using namespace Rcpp;

// [[Rcpp::export]]
NumericMatrix get_ocs_cpp(NumericMatrix df, int n, double c, NumericVector an_pri, double f_c, double a
{
  int rows = df.nrow();
  NumericMatrix results(rows, 2);
  for(int i=0; i<rows; i++){
    double phi_f = df(i,0);
    double phi_a = df(i,1);
    double f = R::rbinom(2*n, phi_f);
    double a = R::rbinom(n, phi_a);
    double post_f = 1- R::pbeta(f_c, an_pri(0)+f, an_pri(1)+2*n-f, 1, 0);
    double post_a = 1- R::pbeta(a_c, an_pri(2)+a, an_pri(3)+n-a, 1, 0);
    results(i,0) = post_f*post_a;
    int green = 1;
    if(phi_f < f_c || phi_a < a_c) green = 0;
    results(i,1) = green;
  }
  return results;
}
```

```
library(Rcpp)
Rcpp::sourceCpp('U:/Projects/MRC SDF/WP2/Papers/cpp_funcs.cpp')
```

```r
# Full MC approximation, working in c++
get_ocs2 <- function(c, n, N, f_c, a_c, des_pri, an_pri=des_pri)
{
  df <- data.frame(p_f = rbeta(N, des_pri[1], des_pri[2]), p_a = rbeta(N, des_pri[3], des_pri[4]))
  r <- get_ocs_cpp(as.matrix(df), n, c, an_pri, f_c, a_c)

  c(sum((r[,1] > c)[!as.logical(r[,2])])/N, sum((r[,1] < c)[as.logical(r[,2])])/N)
}

# For example,
get_ocs2(c=0.5, n=30, N=10^6, f_c=0.8, a_c=0.7, des_pri=c(f_a, f_b, a_a, a_b), an_pri=rep(1,4))
```

```
## [1] 0.056044 0.140224
```

Evaluation

For a number of values of the loss parameter c, we can show how the error rates improve with increasing the sample size n.

```r
get_plot <- function(c, N)
{
  df <- data.frame(n=seq(10, 100, 2))
  df <- cbind(df, t(sapply(df$n, function(n) get_ocs2(c, n, N=N, f_c=0.8, a_c=0.7, des_pri=c(f_a, f_b, a
  names(df)[2:3] <- c("OC1", "OC2")
  df$L <- df$OC1*c + df$OC2*(1-c)
  df$OC1_w <- 1.96*sqrt(df$OC1*(1-df$OC1)/N)
  df$OC2_w <- 1.96*sqrt(df$OC2*(1-df$OC2)/N)
  df$L_v <- (df$OC1*(1-df$OC1)/N)*c^2 + (df$OC2*(1-df$OC2)/N)*(1-c)^2
  df$L_w <- 1.96*sqrt(df$L_v)

  ggplot() +
    geom_ribbon(data=df, aes(x=n, ymin = OC1 - OC1_w, ymax = OC1 + OC1_w), fill = "grey70") +
    geom_ribbon(data=df, aes(x=n, ymin = OC2 - OC2_w, ymax = OC2 + OC2_w), fill = "grey70") +
    geom_ribbon(data=df, aes(x=n, ymin = L - L_w, ymax = L + L_w), fill = "grey70") +
    geom_line(data=df, aes(n, OC1, linetype="aOC1")) +
    geom_line(data=df, aes(n, OC2, linetype="bOC2")) +
    geom_line(data=df, aes(n, L, linetype="cL")) +
    scale_linetype_manual(values=c("aOC1"=1, "bOC2"=2, "cL"=3),
                          labels=c(expression(OC[1]), expression(OC[2]), "E[Loss]")) +
    ylab("Error probability") + xlab("Sample size")
}

p1 <- get_plot(0.25, 10^6)
p2 <- get_plot(0.5, 10^6)
p3 <- get_plot(0.75, 10^6)

p1 <- p1 + theme_minimal() + theme(legend.title=element_blank()) + ylim(c(0,0.3)) +
  ggtitle(expression(paste(c[1], " = 0.25"))) + theme(plot.title = element_text(hjust = 0.5))
p2 <- p2 + theme_minimal() + theme(legend.title=element_blank()) + ylim(c(0,0.3)) +
  ggtitle(expression(paste(c[1], " = 0.5"))) + theme(plot.title = element_text(hjust = 0.5))
p3 <- p3 + theme_minimal() + theme(legend.title=element_blank()) + ylim(c(0,0.3)) +
  ggtitle(expression(paste(c[1], " = 0.75"))) + theme(plot.title = element_text(hjust = 0.5))
```

```
#ggsave("U:/Projects/MRC SDF/WP2/Papers/Figures/tiga_c025.pdf", p1, height=3, width=3)
#ggsave("U:/Projects/MRC SDF/WP2/Papers/Figures/tiga_c05.pdf", p2, height=3, width=3)
#ggsave("U:/Projects/MRC SDF/WP2/Papers/Figures/tiga_c075.pdf", p3, height=3, width=3)
```

Similarly, for a fixed sample size of $n = 60$ we can vary c and plot the results:

```
df <- data.frame(c=seq(0, 1, 0.02))
N <- 10^6

df <- cbind(df, t(sapply(df$c, function(c) get_ocs2(c, n=60, N=10^6, f_c=0.8, a_c=0.7, des_pri=c(f_a, f
names(df)[2:3] <- c("OC1", "OC2")
df$L <- df$OC1*df$c + df$OC2*(1-df$c)
df$OC1_w <- 1.96*sqrt(df$OC1*(1-df$OC1)/N)
df$OC2_w <- 1.96*sqrt(df$OC2*(1-df$OC2)/N)
df$L_v <- (df$OC1*(1-df$OC1)/N)*df$c^2 + (df$OC2*(1-df$OC2)/N)*(1-df$c)^2
df$L_w <- 1.96*sqrt(df$L_v)

p <- ggplot() +
    geom_ribbon(data=df, aes(x=c, ymin = OC1 - OC1_w, ymax = OC1 + OC1_w), fill = "grey70") +
    geom_ribbon(data=df, aes(x=c, ymin = OC2 - OC2_w, ymax = OC2 + OC2_w), fill = "grey70") +
    geom_ribbon(data=df, aes(x=c, ymin = L - L_w, ymax = L + L_w), fill = "grey70") +
    geom_line(data=df, aes(c, OC1, linetype="aOC1")) +
    geom_line(data=df, aes(c, OC2, linetype="bOC2")) +
    geom_line(data=df, aes(c, L, linetype="cL")) +
    scale_linetype_manual(values=c("aOC1"=1, "bOC2"=2, "cL"=3),
                          labels=c(expression(OC[1]), expression(OC[2]), "E[Loss]")) +
    ylab("Error probability") + xlab(expression(paste("Loss parameter ", c[1]))) +
  theme_minimal() + theme(legend.title=element_blank())

#ggsave("U:/Projects/MRC SDF/WP2/Papers/Figures/tiga_n60.pdf", p, height=9, width=14, units="cm")
```

Illustrative example - REACH

Priors

Samples form the design prior distributions described in the paper can be generated using the following function.

```
library(invgamma)

p_sample <- function()
{
  # Generate a sample from the joint prior

  # variance in cluster size (note using factor of k=12)
  alpha <- 20; beta <- 39; nu <- 6; mu0 <- 10
  cl_var <- rinvgamma(1, shape=alpha, rate=beta)
  # mean cluster size
  cl_m <- rnorm(1, mu0, sqrt(cl_var/nu))

  # adherance probability
  ad_m <- 0.9; ad_n <- 30
  ad <- rbeta(1, ad_m*(ad_n+2), (ad_n+2)*(1-ad_m))
  # follow-up probability
  fu_m <- 0.7; fu_n <- 30
  fu <- rbeta(1, fu_m*(fu_n+2), (fu_n+2)*(1-fu_m))

  # effect size
  # follow Spiegelhalter 2001 and assume the ICC and the between-patient variance are independant,
  # putting priors on each of these.
  # ICC
  rho_m <- 0.05; rho_n <- 30
  rho <- rbeta(1, rho_m*(rho_n+2), (rho_n+2)*(1-rho_m))
  # between patient variance, inverse gamme
  var_w <- rinvgamma(1, shape=50, rate=45)
  # effect
  eff <- rnorm(1, 0.2, 0.1)

  return(c(cl_var, cl_m, ad, fu, rho, var_w, eff))
}

# For example,
p_sample()
```

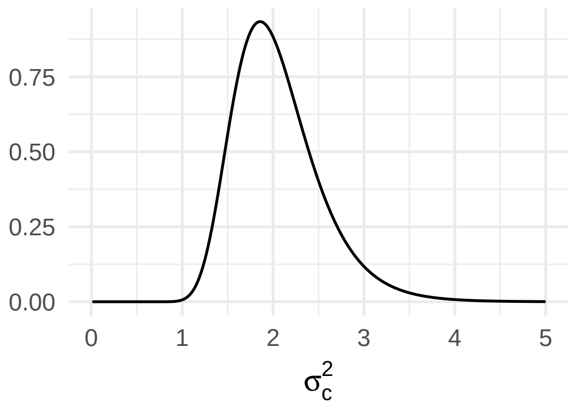
```
## [1]  1.99697170 10.21336213  0.91398221  0.63466233  0.03501252  0.82373261
## [7]  0.21748805
```

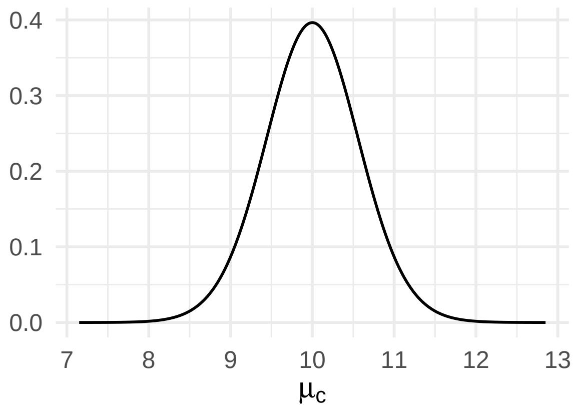
Plotting each of the margincal priors. First, cluster size and follow-up rate:

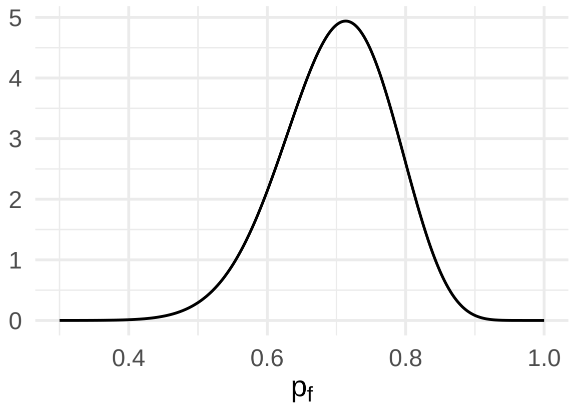
```
library(gridExtra)

# Cluster mean - variance
x <- seq(0,5,0.01)
p <- dinvgamma(x, shape=20, rate=39)
df <- data.frame(v=x, p = p)
p1 <- ggplot(df, aes(v, p)) + geom_line() + theme_minimal() + ylab("") +
  xlab(expression(sigma[c]^2)) + xlim(c(0, 5))

# Cluster mean - mean
# Normal inverse gamma, so sqrt( (alpha*nu)/beta )(x - mu) ~ t_(2*alpha)
x <- seq(-5,5,0.01)
ts <- dt(x, df=2*20)
df <- data.frame(cl = sqrt(39/(20*6))*x+10, p = ts)
p2 <- ggplot(df, aes(cl, p)) + geom_line() + theme_minimal() + ylab("") +
  xlab(expression(mu[c]))

# Follow-up rate
fu <- seq(0,1, 0.001)
p <- dbeta(fu, 22.4, 9.6)
df <- data.frame(fu=fu, p=p)
p3 <- ggplot(df, aes(fu, p)) + geom_line() + ylab("") + theme_minimal() +
  xlab(expression(p[f])) + xlim(c(0.3, 1))

grid.arrange(p1, p2, p3, ncol=2)
```

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

```
## Warning: Removed 300 rows containing missing values (geom_path).
```





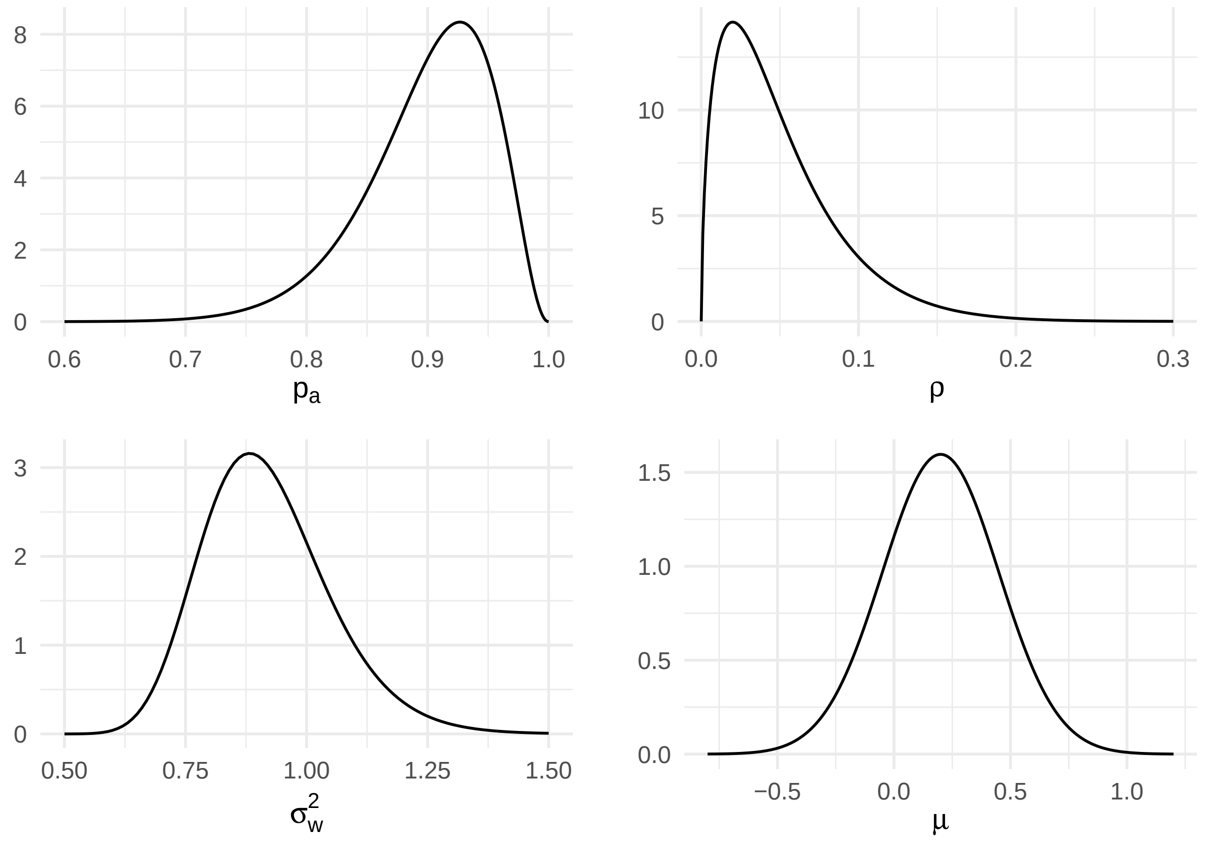


Next, adherence and efficacy:

```
# Adherance

ad <- seq(0,1, 0.001)
p <- dbeta(fu, 28.8, 3.2)
df <- data.frame(ad=ad, p=p)
p4 <- ggplot(df, aes(ad, p)) + geom_line() + theme_minimal() + ylab("") +
  xlab(expression(p[a])) + xlim(c(0.6, 1))

# Efficacy

# ICC
rho_m <- 0.05; rho_n <- 30
rho <- seq(0,1,0.001)
p <- dbeta(rho, rho_m*(rho_n+2), (rho_n+2)*(1-rho_m))
df <- data.frame(rho=rho, p=p)
p5 <- ggplot(df, aes(rho, p)) + geom_line() + theme_minimal() + ylab("") +
  xlab(expression(rho)) + xlim(c(0, 0.3))

# between patient variance, inverse gamma
x <- seq(0,5,0.01)
p <- dinvgamma(x, shape=50, rate=45)
df <- data.frame(v=x, p = p)
p6 <- ggplot(df, aes(v, p)) + geom_line() + theme_minimal() + ylab("") +
  xlab(expression(sigma[w]^2)) + xlim(c(0.5, 1.5))
```

```
# effect
x <- seq(-0.8, 1.2, 0.001)
p <- dnorm(x, 0.2, 0.25)
df <- data.frame(eff=x, p = p)
p7 <- ggplot(df, aes(eff, p)) + geom_line() + theme_minimal() + ylab("") +
  xlab(expression(mu))

grid.arrange(p4, p5, p6, p7, ncol=2)
```

```
## Warning: Removed 600 rows containing missing values (geom_path).

## Warning: Removed 700 rows containing missing values (geom_path).

## Warning: Removed 400 rows containing missing values (geom_path).
```



For the weakly informative analysis, flat $Beta(1,1)$ priros were used for follow-up and adherence rates; for cluster size, the variance prior was $\Gamma^{-1}(1,2)$ and the mean $N(10,10)$; and for efficacy, the ICC prior was a flat $Beta(1,1)$, within-cluster variance was $\Gamma^{-1}(1,2)$, and treatment effect was $N(0.2,1)$.

Hypotheses

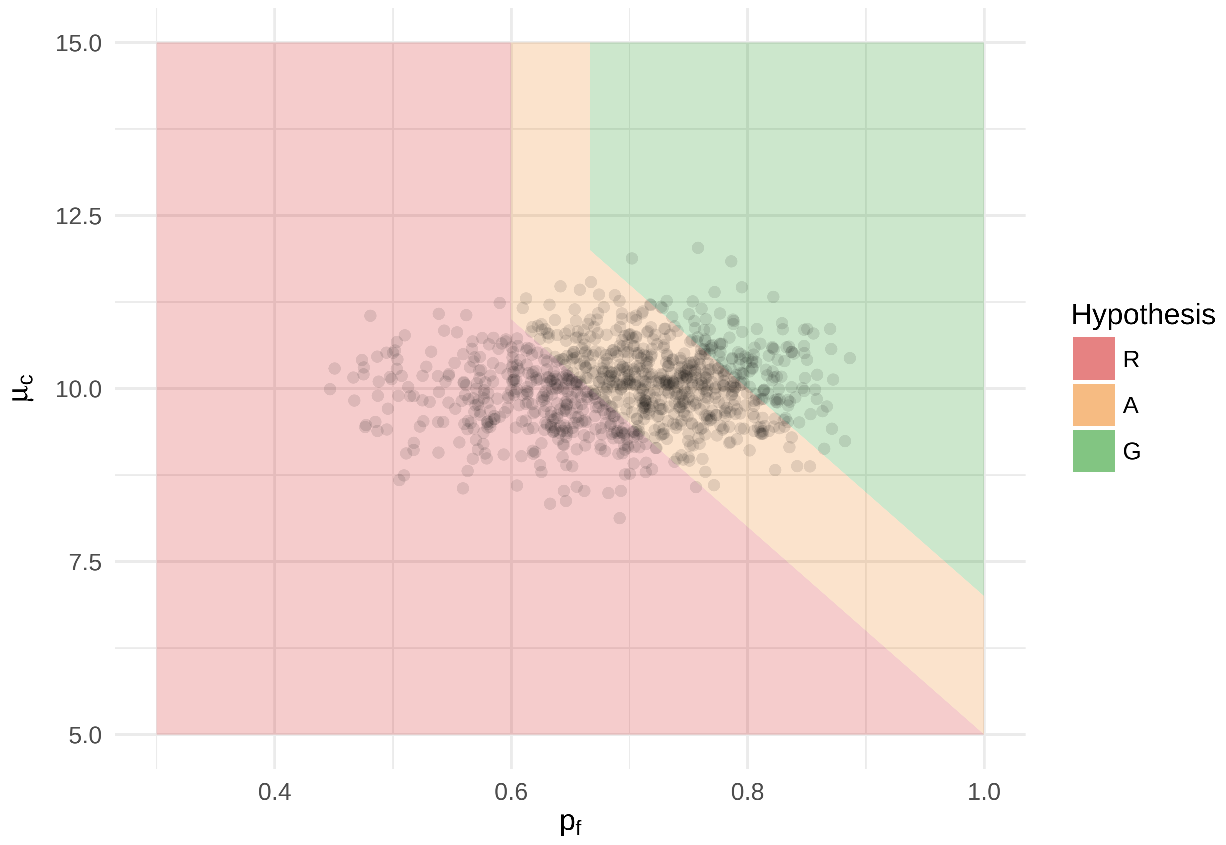
The following function accpets a point in the parameter space and identifies which hypothesis it belongs to, using the hypothesis definitions given in the paper.

```
get_h <- function(x)
{
  # Given a vector of non-nuisance parameters, return the true hypothesis
  cl <- x[1]; ad <- x[2]; fu <- x[3]; eff <- x[4]

  # Follow-up and cluster size
  go1 <- 1
  if(fu < 0.6666666 | (22-15*fu > cl) ) go1 <- 0
  if(fu < 0.6 | (20-15*fu > cl) ) go1 <- -1

  # Adherance and efficacy
  go2 <- 1
  if(ad < 0.6 | (1.05714286-0.5714286*eff > ad) ) go2 <- 0
  if(ad < 0.5 | (0.9571429-0.5714286*eff > ad) ) go2 <- -1

  go <- 1
  if(go1==0 | go2==0) go <- 0
  if(go1==-1 | go2==-1) go <- -1

  return(c(go1, go2, go))
}
```

We can visualise the hypotheses as follows: Plot the hypotheses:

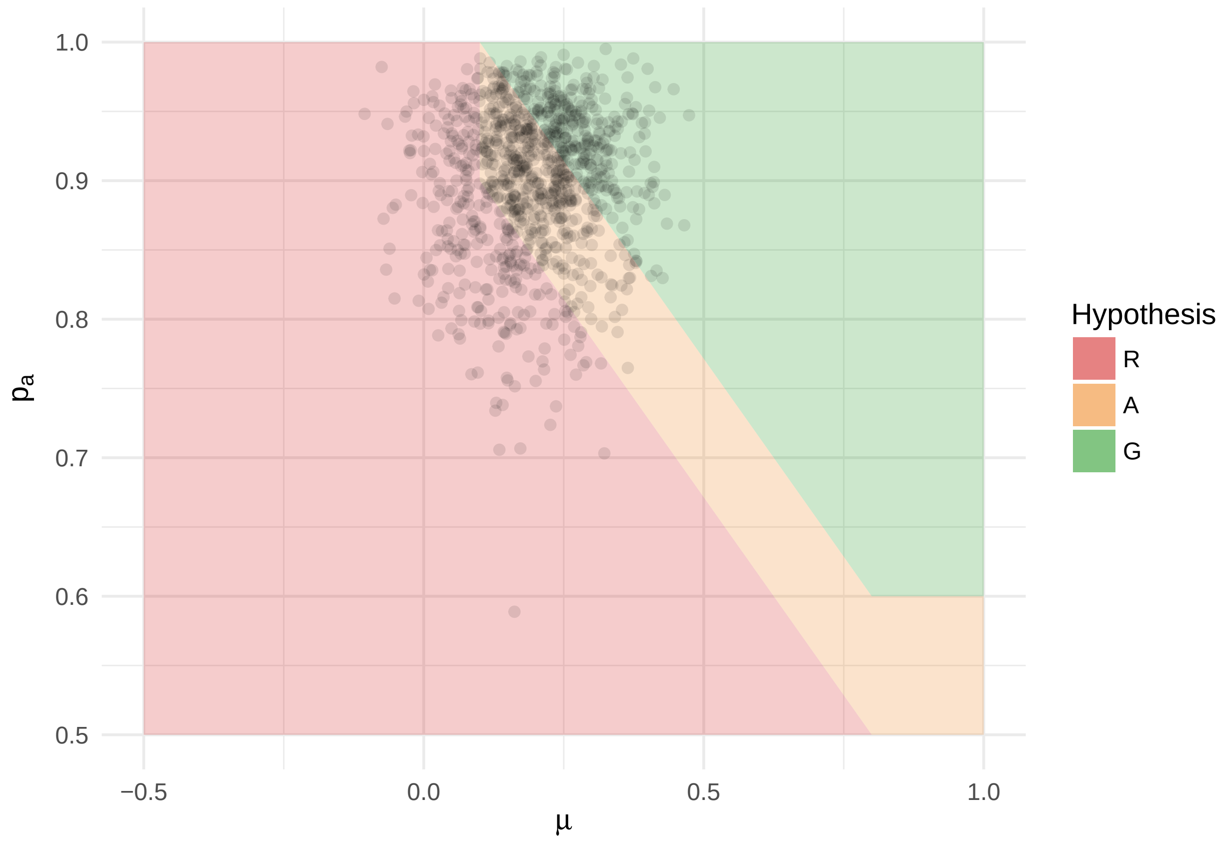
```
draw_hyp <- function(z, lim, df)
{
  # Draw a 2-D hypotheses and overlay with prior samples

  # z = vector of boundary points, in order as specified below
  # lim = matrix of limits
  # df = prior sample

  x_ra <- z[1]; x_ag <- z[2]
  y_ra <- z[3]; y_ag <- z[4]
  x_ra_c1 <- z[5]; y_ra_c1 <- z[6]
  x_ra_c2 <- z[7]; y_ra_c2 <- z[8]
  x_ag_c1 <- z[9]; y_ag_c1 <- z[10]
  x_ag_c2 <- z[11]; y_ag_c2 <- z[12]

  df_polr <- data.frame(x=c(lim[1,1], lim[1,1], x_ra, x_ra_c1, x_ra_c2, lim[1,2], lim[1,2]),
                      y=c(lim[2,1], lim[2,2], lim[2,2], y_ra_c1, y_ra_c2, y_ra, lim[2,1]),t=-1)

  df_pola <- data.frame(x=c(x_ra, x_ag, x_ag_c1, x_ag_c2, lim[1,2], lim[1,2],
                          x_ra_c2, x_ra_c1),
                      y=c(lim[2,2], lim[2,2], y_ag_c1, y_ag_c2, y_ag, y_ra,
                          y_ra_c2, y_ra_c1),t=0)

  df_polg <- data.frame(x=c(x_ag, x_ag_c1, x_ag_c2, lim[1,2], lim[1,2]),
                      y=c(lim[2,2], y_ag_c1, y_ag_c2, y_ag, lim[2,2]),t=1)

  p <- ggplot(df, aes(x, y)) +
    geom_polygon(data=df_polr, alpha=0.2, aes(fill=as.factor(t))) +
    geom_polygon(data=df_pola, alpha=0.2, aes(fill=as.factor(t))) +
    geom_polygon(data=df_polg, alpha=0.2, aes(fill=as.factor(t))) +
    geom_point(alpha=0.1) +
    scale_fill_manual(name="Hypothesis",
                      labels=c("R", "A", "G"),
```

```r
                      values=c("red3", "darkorange2", "green4")) +
    theme_minimal()

  return(p)
}


# Get some prior samples
df <- data.frame(t(replicate(1000, p_sample()))) # c(cl_var, cl_m, ad, fu, rho, var_w, eff))

# Follow up and cluster size

# Define the marginal boundaries
fu_ra <- 0.6; fu_ag <- 0.66666667
cl_ra <- 5; cl_ag <- 7

# Define the combined boundary points
fu_ra_c1 <- 0.6; cl_ra_c1 <- 11
fu_ra_c2 <- 1; cl_ra_c2 <- 5
fu_ag_c1 <- 0.66666667; cl_ag_c1 <- 12
fu_ag_c2 <- 1; cl_ag_c2 <- 7

z <- c(fu_ra, fu_ag, cl_ra, cl_ag,
       fu_ra_c1, cl_ra_c1,
       fu_ra_c2, cl_ra_c2,
       fu_ag_c1, cl_ag_c1,
       fu_ag_c2, cl_ag_c2)

lim <- matrix(c(0.3, 1, 5, 15), ncol=2, byrow = T)

df2 <- df[,c(4,2)]
names(df2) <- c("x", "y")

# Draw plot
p <- draw_hyp(z, lim, df2)
p + ylab(expression(mu[c])) + xlab(expression(p[f]))
```



```
#ggsave("../Papers/Figures/hyp_fu_cl.pdf", width = 10, height = 7, units="cm")

# Efficacy and adherance

# Define the marginal boundaries
eff_ra <- 0.1; eff_ag <- 0.1
ad_ra <- 0.5; ad_ag <- 0.6

# Define the combined boundary points
eff_ra_c1 <- 0.1; ad_ra_c1 <- 0.9
eff_ra_c2 <- 0.8; ad_ra_c2 <- 0.5
eff_ag_c1 <- 0.1; ad_ag_c1 <- 1
eff_ag_c2 <- 0.8; ad_ag_c2 <- 0.6

z <- c(eff_ra, eff_ag, ad_ra, ad_ag,
       eff_ra_c1, ad_ra_c1,
       eff_ra_c2, ad_ra_c2,
       eff_ag_c1, ad_ag_c1,
       eff_ag_c2, ad_ag_c2)

lim <- matrix(c(-0.5, 1, 0.5, 1), ncol=2, byrow = T)

df3 <- df[,c(7,3)]
names(df3) <- c("x", "y")

# Draw plot
```

```r
p <- draw_hyp(z, lim, df3)
p + ylab(expression(p[a])) + xlab(expression(mu))
```



```r
#ggsave("../Papers/Figures/hyp_ad_eff.pdf", width = 10, height = 7, units="cm")
```

Simulation model

```r
sim_trial <- function(i)
{
  # Sample all parameters from the prior
  p <- p_sample()
  cl_var <- p[1]; cl_m <- p[2]; ad <- p[3]; fu <- p[4]; rho <- p[5]; var_w <- p[6]; eff <- p[7]

  # Determine the true hypothesis
  hs <- get_h(c(cl_m, ad, fu, eff))
  h1 <- hs[1]; h2 <- hs[2]; h <- hs[3]

  # Simulate pilot data and calculate summary statistics
  k <- 12
  data <- matrix(seq(1,k), ncol=1)

  # Randomise care homes
  split <- floor(k/2) + rbinom(1,1,(k/2)%%1)
  data <- cbind(data, c(rep(0,split), rep(1,k-split)))
```

```r
# Simulate intervention delivery success at care home level
ad_h <- ad
data <- cbind(data, c(rep(0,split), rbinom(k-split, 1, ad_h)))
ad_c <- sum(data[,3])
ad_n <- k-split
ad_est <- mean(data[(k-split+1):nrow(data),3])

# Simulate recruitment of residents
gen_m <- function(row, cl_m, cl_var)
{
  return(round(rnorm(1, cl_m, sqrt(cl_var))))
}
data <- cbind(data, apply(data, 1, gen_m, cl_m=cl_m, cl_var=cl_var))
#data <- cbind(data, rep(10, 12))
data[,4] <- ifelse(data[,4] < 1, 1, data[,4])
cl_m_est <- mean(data[,4])
cl_var_est <- var(data[,4])
cl_n <- nrow(data)

# Simulate loss to follow-up
lose <- function(row, fu){
  return(rbinom(1, row[4], fu))
}
data <- cbind(data, apply(data, 1, lose, fu=fu))
data[,5] <- ifelse(data[,5] < 1, 1, data[,5])
fu_est <- sum(data[,5])/sum(data[,4])
fu_c <- sum(data[,5])
fu_n <- sum(data[,4])

# Simulate cluster effects
var_b <- rho*var_w/(1-rho)
var_t <- var_b + var_w
data <- cbind(data, rnorm(k,0,sqrt(var_b)))
b_sd <- sd(data[,6])

# Add treatment effects
d <- eff
data <- cbind(data, d*as.numeric(data[,2] == 1 & data[,3] == 1))

# Simulate patient outcomes
data <- data[rep(seq_len(nrow(data)), data[,5]),]
gen_y <- function(row, var_w){
  return(rnorm(1, row[6] + row[7], sqrt(var_w)))
}
data <- cbind(data, apply(data, 1, gen_y, var_w=var_w))
d_est <- mean(data[data[,2] == 1,8])-mean(data[data[,2] == 0,8])
t_sd <- sd(data[,8])

data <- cbind(data, data[,3])
data <- cbind(data, data[,8] - (!data[,9])*eff)
d_est <- mean(data[data[,2] == 1,10])-mean(data[data[,2] == 0,10])

ests <- c(h1, h2, h, ad_est, cl_m_est, fu_est, d_est, b_sd, t_sd, cl_m_est, cl_n, cl_var_est, fu_c, f
  return(c(ests, cl_m, fu, ad, eff))
}
```

Analysis

For a simulated data set, we use rstan to conduct the MCMC analysis.

```
library("rstan")

#res1 <- res2 <- res3 <- NULL
preds <- NULL
for(i in 1:10000){
  print(i)

  d <- sim_trial(i)

  to_add <- data.frame(t(d[[1]]))
  names(to_add) <- letters[1:(ncol(to_add))]

  data <- d[[2]]
  h <- d[[1]][1]

  # Reduce data to first in each cluster to get follow up
  red <- data[!duplicated(data[,1]),]
  # Make data
  REACH_data <- list(K = 12,
                     M = red[,4],
                     clus = data[,1],
                     N = nrow(data),
                     y = data[,10],
                     # Patient level adherance
                     a_p = data[,9],
                     # Cluster level adherance
                     a_c = red[red[,2] == 1, 3],
                     lost = sum(red[,4]) - sum(red[,5]),
                     trt = data[,2])

  # Fit the model
  # WP2_ex_REACH_5_... 1 - totally noninformative, vague priors on everything
  #                    2 - vauge priors on substantive params only
  #                    3 - vauge priors on substantive params except adherance (at cluster level)
  # When not using a vauge prior, we use the design prior

  fit <- stan(file = 'WP2_ex_REACH_5_1.stan', data = REACH_data,
              iter = 5000, chains = 4, control=list(adapt_delta =0.9))

  pr <- summary(fit)$summary["pr",c("mean", "se_mean")]
  pa <- summary(fit)$summary["pa",c("mean", "se_mean")]
  pg <- summary(fit)$summary["pg",c("mean", "se_mean")]
  to_add1 <- cbind(to_add, t(c(h, pr, pa, pg)))
  names(to_add1)[(ncol(to_add1)-6):ncol(to_add1)] <- c("y", "pr", "pr_se", "pa", "pa_se", "pg", "pg_se"
  res1 <- rbind(res1, to_add1)
```

```r
  fit2 <- stan(file = 'WP2_ex_REACH_5_2.stan', data = REACH_data,
              iter = 5000, chains = 4, control=list(adapt_delta =0.9))

  pr <- summary(fit2)$summary["pr",c("mean", "se_mean")]
  pa <- summary(fit2)$summary["pa",c("mean", "se_mean")]
  pg <- summary(fit2)$summary["pg",c("mean", "se_mean")]
  to_add2 <- cbind(to_add, t(c(h, pr, pa, pg)))
  names(to_add2)[(ncol(to_add2)-6):ncol(to_add2)] <- c("y", "pr", "pr_se", "pa", "pa_se", "pg", "pg_se"
  res2 <- rbind(res2, to_add2)

  fit3 <- stan(file = 'WP2_ex_REACH_5_3.stan', data = REACH_data,
              iter = 5000, chains = 4, control=list(adapt_delta =0.9))

  pr <- summary(fit3)$summary["pr",c("mean", "se_mean")]
  pa <- summary(fit3)$summary["pa",c("mean", "se_mean")]
  pg <- summary(fit3)$summary["pg",c("mean", "se_mean")]
  to_add3 <- cbind(to_add, t(c(h, pr, pa, pg)))
  names(to_add3)[(ncol(to_add3)-6):ncol(to_add3)] <- c("y", "pr", "pr_se", "pa", "pa_se", "pg", "pg_se"
  res3 <- rbind(res3, to_add3)
}

df <- as.data.frame(extract(fit, c("d", "d_rho", "c_m", "c_sigsq", "d_sigsq_w", "p_f", "p_a")))

#saveRDS(res1, "exp_1.Rda")
#saveRDS(res2, "exp_2.Rda")
#saveRDS(res3, "exp_3.Rda")
```

The Stan files are of the same form but with different analysis priors, as described previously. The weakly informative model is as follows:

```stan
data {
  int<lower=0> K;  // # of clusters
  real<lower=0> M[K]; // # array of cluster sizes
  int<lower=0> N;  // total sample size
  int<lower=0> clus[N]; // cluster index allocations
  vector[N] y;  // responses
  int<lower=0,upper=1> a_p[N]; // adherences - patient level
  int<lower=0,upper=1> a_c[6]; // adherences - cluster level
  int<lower=0,upper=1> trt[N]; // treatment allocation
  int<lower=0> lost;  // number missed in follow-up
}
parameters {
  real<lower=0,upper=1> p_f; // prob of being followed-up
  real<lower=0,upper=1> p_a;  // prob of adhering to treatment
  real<lower=0> c_m; // Mean cluster size
  real<lower=0> c_sigsq; //cluster size variance
  real d; // Mean treatment effect
  real<lower=0,upper=1> d_rho;  // ICC for treatment effect
  real<lower=0> d_sigsq_w;  // whithin cluster variance for treatment effect

  vector[K] u;  // random effects
}
transformed parameters {
  real<lower=0> c_sig;
  real<lower=0> d_sig_w;
  real<lower=0> d_sig_b;

  c_sig = sqrt(c_sigsq);
  d_sig_w = sqrt(d_sigsq_w);
  d_sig_b = sqrt(d_rho*d_sigsq_w/(1-d_rho));
}
model {
  // Priors
  p_f ~ beta(1, 1);
  p_a ~ beta(1, 1);
  c_sigsq ~ inv_gamma(1, 2);
  c_m ~ normal(10, 10);

  d_rho ~ beta(1, 1);
  d_sigsq_w ~ inv_gamma(1, 2);
  d ~ normal(0.2, 1);

  u ~ normal(0, d_sig_b);

  // Follow-up
  N ~ binomial(N+lost, p_f);

  // Cluster size
  M ~ normal(c_m, c_sig);

  // Adherance
  a_c ~ bernoulli(p_a);

  for(i in 1:N){
    // Efficacy
    y[i] ~ normal(trt[i]*a_p[i]*d + u[clus[i]], d_sig_w);
  }
}

generated quantities {
  real<lower=0,upper=1> pr;
  real<lower=0,upper=1> pa;
  real<lower=0,upper=1> pg;

  pg = !(p_f < 0.6666666 || (22-15*p_f > c_m) || p_a < 0.6 || (1.05714286-0.5714286*d > p_a));
  pr = (p_f < 0.6 || (20-15*p_f > c_m) || p_a < 0.5 || (0.9571429-0.5714286*d > p_a));
  pa = !(pr == 1 || pg == 1);
}
```

Operating characteristics

Generate the set of loss parameter vectors to be evaluated.

```r
library(lhs)
```

```
## Warning: package 'lhs' was built under R version 3.4.4
```

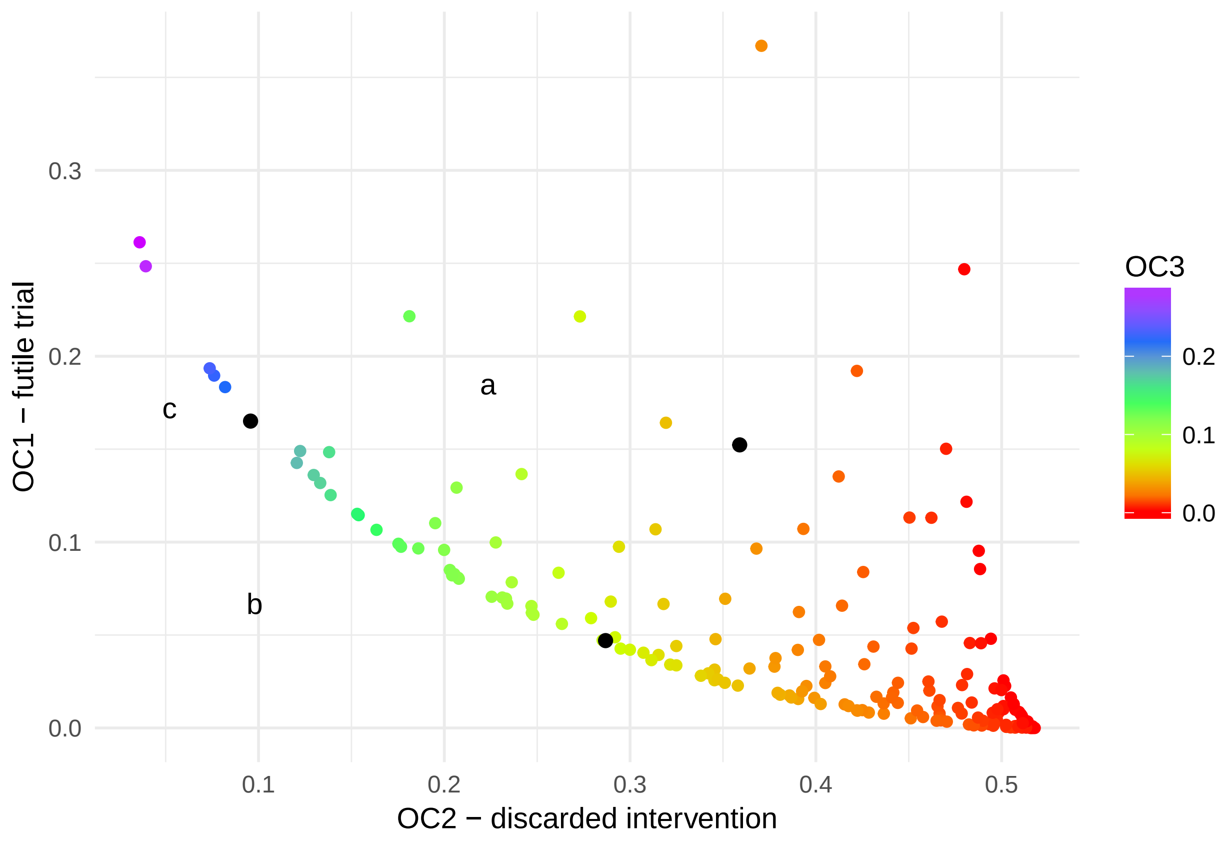
```r
set.seed(19832)
d <- as.data.frame(maximinLHS(500, 2)); names(d) <- c("c1", "c2")
d <- d[(d$c1+d$c2)<1,]
```

Next, we load the results of the nested MC analysis. Note here that we carried out three analyses, each using the same set of simulated (ϕ, x) but changing what analysis priors were used. The file `exp_i.Rda` is associated with the Stan script 'WP2_ex_REACH_5_i.stan' containing the full details of the analysis. The cases i = 1,2,3 correspond to weakly informative priors, informative priors for nuisance parameters only, and informative priors for nuisance parameters and for the probability of adherence.

```r
# exp_i.Rda orresponding to WP2_ex_REACH_5_i.stan
# 1 - weakly informative
# 2 - informative nuisance
# 3 - informative nuisance plus adherence
res <- readRDS("../R/exp_1.Rda")
#res <- readRDS("../R/exp_2.Rda")
#res <- readRDS("../R/exp_3.Rda")
```

Now we estimate the operating characteristics for each loss parameter vector, discard all those which are dominated, and plot the results.

```r
library(mco)

get_decision <- function(probs, r)
{
  # r = loss paramaters c(c_1, c_2, c_3)
  # probs = posterior probabilities (p_R, p_G)

  u_r <- (1-sum(probs))*r[3] + probs[2]*r[3]
  u_a <- probs[1]*r[1] + probs[1]*r[2] + probs[2]*r[2]
  u_g <- probs[1]*r[1] + (1-sum(probs))*r[1] + (1-sum(probs))*r[3]

  return(which.min(c(u_r,u_a,u_g))-2)
}

get_objectives <- function(rule, res)
{
  rule <- c(rule, 1-sum(rule))
  # rule = loss parameters (c_1, c_2, c_3)

  # Get the decisions for each simulated data set and posterior analysis
  dec <- apply(res[,c("pr", "pg")], 1, get_decision, r=rule)
  res$dec <- dec

  # Calculate the proportion of samples in each decision x hypothesis cell
  n <- nrow(res)
  p_0m1 <- nrow(res[res$dec==0 & res$a == -1,])/n
  p_1m1 <- nrow(res[res$dec==1 & res$a == -1,])/n
  p_10 <- nrow(res[res$dec==1 & res$a == 0,])/n
  p_01 <- nrow(res[res$dec==0 & res$a == 1,])/n
  p_m10 <- nrow(res[res$dec==-1 & res$a == 0,])/n
  p_m11 <- nrow(res[res$dec==-1 & res$a == 1,])/n

  # Get operating characteristics
  # i) Futile trial
  OC1 <-  p_0m1 + p_1m1 + p_10
  # ii) Unnecesary adjustments
  OC2 <-  p_0m1 + p_01
  # iii) Discarding a promising intervention
  OC3 <-  p_m10 + p_m11 + p_10
  # Expected loss
  EL <- rule[3]*p_m10 + rule[3]*p_m11 +
        (rule[1]+rule[2])*p_0m1 + rule[2]*p_01 +
        rule[1]*p_1m1 + (rule[1]+rule[3])*p_10

  return(c(OC1, OC2, OC3, EL))
}

rs <- t(apply(d, 1, get_objectives, res=res))

# Filter out dominated parameters
pf_r <- paretoFilter(rs[,1:3])

df <- as.data.frame(cbind(rs[row.names(pf_r),], d[row.names(pf_r),]))
names(df) <- c("OC1", "OC3", "OC2", "Eu", "c1", "c2")
shift <- 0.02
# vector of solution indices to highlight
hi <- c(131,49,45) # for res1

ggplot(df, aes(OC2, OC1, colour=OC3)) + geom_point() +
  scale_color_gradientn(colours = rainbow(5)) + theme_minimal() +
  ylab("OC1 - futile trial") + xlab("OC2 - discarded intervention") +
  coord_fixed() +
  geom_point(data=df[hi,], colour="black", size=2) +
  annotate("text", x = df[hi[1],"OC3"]+shift, y = df[hi[1],"OC1"]+shift, label = "a") +
  annotate("text", x = df[hi[2],"OC3"]+shift, y = df[hi[2],"OC1"]+shift, label = "b") +
  annotate("text", x = df[hi[3],"OC3"]+shift, y = df[hi[3],"OC1"]+shift, label = "c")
```



```r
#ggsave("./Figures/p_front.pdf", width = 14, height = 9, units="cm")
```

For the three points highligherd in the figure, print their actual operating characteristics:

```r
round(df[hi,], 3)

p <- 0.165; sqrt(p*(1-p)/(10^4))
```

Plot the loss parameter values against the obtained operating characteristics:

```r
library(reshape2)

g_df <- df[,c(1,2,3,5,6)]
g_df$c3 <- g_df$c2
g_df$c2 <- 1 - g_df$c1 - g_df$c3
g_df <- melt(g_df, id=c("OC1", "OC2", "OC3"))
names(g_df)[4:5] <- c("cost_t", "cost")
g_df <- melt(g_df, id=c("cost_t", "cost"))
names(g_df)[3:4] <- c("OC_t", "prob")
g_df$cost_t2 <- factor(g_df$cost_t, labels=c(1,2,3))
g_df$OC_t2 <- factor(g_df$OC_t, labels=c(1,2,3))

ggplot(g_df, aes(cost, prob)) + geom_point(alpha=0.5) +
  facet_grid(OC_t2 ~ cost_t2, labeller = label_bquote(rows = OC[.(OC_t2)], cols = c[.(cost_t2)]), switc
  theme_minimal() +
  xlab("Cost") + ylab("Probability")
```

```
#ggsave("./Figures/cost_OCs.pdf", width = 14, height = 9, units="cm")
```

Compare the results of the three different types of analysis priors:

```
# Comparison of different analysis priors

# Weakly informative everywhere
res <- readRDS("../R/exp_1.Rda")
rs <- t(apply(d, 1, get_objectives, res=res))
df1 <- as.data.frame(cbind(rs, d))
names(df1) <- c("OC1", "OC2", "OC3", "EL", "c1", "c2")
df1$t <- "WI"

# Informative nuisance parameters
res <- readRDS("../R/exp_2.Rda")
rs <- t(apply(d, 1, get_objectives, res=res))
df2 <- as.data.frame(cbind(rs, d))
names(df2) <- c("OC1", "OC2", "OC3", "EL", "c1", "c2")
df2$t <- "IN"

# Informative nuisance and p_a, care home delivery probability
res <- readRDS("../R/exp_3.Rda")
rs <- t(apply(d, 1, get_objectives, res=res))
df3 <- as.data.frame(cbind(rs, d))
names(df3) <- c("OC1", "OC2", "OC3", "EL", "c1", "c2")
df3$t <- "IA"

temp2 <- melt(df1[,c(1,2,3,4,5,6,7)], id.vars = c("c1", "c2", "t"))[,3:5]
temp3 <- melt(df3[,c(1,2,3,4,5,6,7)], id.vars = c("c1", "c2", "t"))[,3:5]
temp <- data.frame(IN = temp2[,3], IA = temp3[,3], OC = temp2[,2])
ggplot(temp, aes(IN, IA, colour = OC, shape=OC)) + geom_point(alpha=0.6) +
  theme_minimal() +
  xlab("Weakly informative") + ylab("Partially informative") +
  scale_colour_discrete(name="",
                        labels=c(expression(OC[1]), expression(OC[2]), expression(OC[3]), "E[L]")) +
  scale_shape_discrete(name="",
                        labels=c(expression(OC[1]), expression(OC[2]), expression(OC[3]), "E[L]")) +
  geom_segment(x = 0, xend = 1, y = 0, yend = 1, linetype=2, colour="grey")
```