```
# clear the workspace
rm(list = ls(all = T))

library(rjags)
library(R2OpenBUGS)
library(coda)
library(extraDistr)
library(ggplot2)

##### Function to generate custom summaries from JAGS fits
##### Written by Benjamin Staton, Auburn University
post.summ = function(post, var) {

  # coerce to matrix for easy subsetting
  post.samp = as.matrix(post)

  # if parameter is indexed
  if(substr(var, nchar(var), nchar(var)) == "[") {
    # extract columns with headers equal to the desired variable
    post.sub = post.samp[,substr(colnames(post.samp), 1, nchar(var))
== var]
    # calculate desired quantities
    summ = apply(post.sub, 2, function(x) c(mean = mean(x), sd =
sd(x), quantile(x, c(0.5, 0.025, 0.975))))
    return(summ)
  }

  # if parameter is not indexed
  if(substr(var, nchar(var), nchar(var)) != "[") {
    # extract the column with the same header as the desired variable
    post.sub = post.samp[,substr(colnames(post.samp), 1, nchar(var))
== var]
    # calculate the desired quantities
    summ = c(mean = mean(post.sub), sd = sd(post.sub),
quantile(post.sub, c(0.5, 0.025, 0.975)))
    return(summ)
  }
}

############################################################
#### ESTIMATION OF Q (RATE OF SUCCESS OF EACH SIRE)
#### AND CALCULATION OF M (# MATES) FROM R (# SIRES)
############################################################

##### read and prepare data #####
dat <- read.csv("paternity_mammals.csv")

# expand data to create multiple broods within each species (sample
size * 10)
for(i in 1:nrow(dat)){
```

```
  temp.sp <- expand.grid(species = rep(dat[i,]$species, dat[i,]
$nbrood*10))
  if(i == 1){
    temp <- temp.sp
  } else {
    temp <- rbind(temp, temp.sp)
  }
}
mates.df <- merge(temp, dat, all.x = TRUE, all.y = FALSE)


##### specify model code #####
model_byspecies_q.file <- "model_byspecies_q.txt"
jagsscript.byspecies_q <- cat("
  model{
  # define prior: probability of q (prob of success for each sire)
  q ~ dbeta(1,1)

  # likelihood: stochastic relationship
  # sires ~ (truncated binomial)
  for(i in 1:N){
    nsires[i] ~ dbinom(q, littersize[i]) T(0,)
  }

  # DERIVED QUANTITIES
  # mates ~ (negative binomial);
  for(i in 1:N){
    failed.m[i] ~ dnbinom(q, nsires[i]) T(nsires[i],)
    nmates[i] <- failed.m[i] + nsires[i]
  }
  for(i in 1:N){
    est.M[i] <- nsires[i]/q
  }

  # Calculated quantities from original data
  for(j in 1:J){
    avg.R[j] <- avgbrood[j]*q/(1-(1-q)^avgbrood[j])     # avg sires,
given avgbrood and estimated q
  }
  for(j in 1:J){
    avg.M[j] <- avgsire[j]/q     # avg mates, given avgsire and
estimated q
  }
  for(j in 1:J){
    est.pmult.kq[j] <- (1 - (avgbrood[j] * q) * (1 - q)^(avgbrood[j] -
1)) / (1 - (1 - q)^(avgbrood[j]))
    # estimated prob. of mult. paternity, given avgbrood and estimated
q
  }
}
```

```
",file = model_byspecies_q.file)



#### BEGINNING OF REGENERATION OF DATA LOOP ####
# Remove any datapoints with NA for avgsire and avgbrood, since
Poisson data gen will fail
mates.dat <- mates.df[!is.na(mates.df$avgsire),]

set.seed(36849)

#### (1) generate broods and sires from Poisson for each species ####
mates <- cbind(mates.dat, littersize = NA, nsires = NA)
for(i in 1:nrow(mates)){
  # generate broods
  nsample <- mates[i,]$nbrood   # number of broods
  avg.bs <- mates[i,]$avgbrood  # average brood size
  gen.k <- rtpois(1, avg.bs, a = 0, b = Inf)
  mates[i,]$littersize <- gen.k
  # generate sires
  avg.sire <- mates[i,]$avgsire  # average number of sires
  sire.max <- mates[i,]$littersize # max number of sires per litter is
litter size
  gen.r <- rtpois(1, avg.sire, a = 0, b = sire.max)
  mates[i,]$nsires <- gen.r
}
## remove an average sire value of NA
mates <- mates[!is.na(mates$avgsire), ]


##### (2) MCMC dimensions #####
ni = 10000
nb = 1000
nc = 2 # needs to match number of initial values proposed
nt = 2
n.iter = ni + nb

##### (3) parameters to monitor #####
params = c("q", "nsires", "failed.m", "nmates", "est.M", "avg.R",
"avg.M", "est.pmult.kq")

##### list to hold results from JAGS
jags.results <- list()

##### (4) run the model in JAGS #####
# Initial conditions for loop
# Likelihood needs to change for each nsire value, so q calculated for
each # nsire
mates.p <- mates
species <- sort(unique(mates.p$species))
```

```
b <- 1
# Start JAGS
for(r in species){
  # data containing only relevant values
  m.dat <- mates.p[mates.p$species==r,]
  orig.dat <- dat[dat$species==r,]
  jags.dat <- list(nsires = m.dat$nsires,
                   littersize = m.dat$littersize,
                   N = nrow(m.dat),
                   avgbrood = orig.dat$avgbrood,
                   avgsire = orig.dat$avgsire,
                   J = nrow(orig.dat))

  # run JAGS (not using initial values for q)
  jmod <- jags.model(file = model_byspecies_q.file, data = jags.dat,
n.chains = nc, n.adapt = 1000)
  update(jmod, n.iter = nb, by = 1, progress.bar = 'text')
  post <- coda.samples(jmod, params, n.iter = ni, thin = nt)

  # save current JAGS output
  name <- paste0("species=",r)
  jags.results[[name]] <- post

  # Save quantities from data
  MCMC_temp <- cbind(
    as.character(orig.dat$species),
    as.numeric(orig.dat$avgbrood),
    as.numeric(orig.dat$avgsire),
    as.numeric(orig.dat$pmult),
    post.summ(post, "q")[[1]],  # mean of the est. param. q
    post.summ(post, "q")[[2]],  # sd of the est. param. q
    post.summ(post, "q")[[4]],  # 2.5% est. param. q
    post.summ(post, "q")[[5]],  # 97.5% est. param. q
    post.summ(post, "nsires")[[1]],  # mean of nsires
    post.summ(post, "nsires")[[2]],  # sd of nsires
    post.summ(post, "nsires")[[4]],  # 2.5% nsires
    post.summ(post, "nsires")[[5]],  # 97.5% nsires
    post.summ(post, "nmates")[[1]],  # mean of nmates
    post.summ(post, "nmates")[[2]],  # sd of nmates
    post.summ(post, "nmates")[[4]],  # 2.5% nmates
    post.summ(post, "nmates")[[5]],  # 97.5% nmates
    post.summ(post, "est.M")[[1]],  # mean of est.M
    post.summ(post, "est.M")[[2]],  # sd of est.M
    post.summ(post, "est.M")[[4]],  # 2.5% est.M
    post.summ(post, "est.M")[[5]],  # 97.5% est.M
    post.summ(post, "est.pmult.kq")[[1]],  # mean of the est.pmult.kq
    post.summ(post, "est.pmult.kq")[[2]],  # sd of the est.pmult.kq
    post.summ(post, "est.pmult.kq")[[4]],  # 2.5% est.pmult.kq
    post.summ(post, "est.pmult.kq")[[5]],  # 97.5% est.pmult.kq
    post.summ(post, "avg.R")[[1]],  # mean of avg.R
```

```
    post.summ(post, "avg.R")[[2]],  # sd of avg.R
    post.summ(post, "avg.R")[[4]],  # 2.5% avg.R
    post.summ(post, "avg.R")[[5]],  # 97.5% avg.R
    post.summ(post, "avg.M")[[1]],  # mean of avg.M
    post.summ(post, "avg.M")[[2]],  # sd of avg.M
    post.summ(post, "avg.M")[[4]],  # 2.5% avg.M
    post.summ(post, "avg.M")[[5]]  # 97.5% avg.M
    )

if(b==1) MCMC_sumtemp <- MCMC_temp else MCMC_sumtemp <-
rbind(MCMC_sumtemp, MCMC_temp)

# move to next step
b <- b + 1
}

MCMC_summary <- data.frame(species = as.character(MCMC_sumtemp[,1]),
                           avgbrood = as.numeric(MCMC_sumtemp[,2]),
                           avgsire = as.numeric(MCMC_sumtemp[,3]),
                           pmult = as.numeric(MCMC_sumtemp[,4]),
                           mean.q = as.numeric(MCMC_sumtemp[,5]),
                           sd.q = as.numeric(MCMC_sumtemp[,6]),
                           q2.5 = as.numeric(MCMC_sumtemp[,7]),
                           q97.5 = as.numeric(MCMC_sumtemp[,8]),
                           mean.nsires = as.numeric(MCMC_sumtemp[,
9]),
                           sd.nsires = as.numeric(MCMC_sumtemp[,
10]),
                           nsires2.5 = as.numeric(MCMC_sumtemp[,
11]),
                           nsires97.5 = as.numeric(MCMC_sumtemp[,
12]),
                           mean.nmates = as.numeric(MCMC_sumtemp[,
13]),
                           sd.nmates = as.numeric(MCMC_sumtemp[,
14]),
                           nmates2.5 = as.numeric(MCMC_sumtemp[,
15]),
                           nmates97.5 = as.numeric(MCMC_sumtemp[,
16]),
                           mean.estM = as.numeric(MCMC_sumtemp[,
17]),
                           sd.estM = as.numeric(MCMC_sumtemp[,18]),
                           estM2.5 = as.numeric(MCMC_sumtemp[,19]),
                           estM97.5 = as.numeric(MCMC_sumtemp[,20]),
                           mean.est.pmult =
as.numeric(MCMC_sumtemp[,21]),
                           sd.est.pmult = as.numeric(MCMC_sumtemp[,
22]),
                           est.pmult2.5 = as.numeric(MCMC_sumtemp[,
```

```
23]),
                              est.pmult97.5 = as.numeric(MCMC_sumtemp[,
24]),
                              mean.avgR = as.numeric(MCMC_sumtemp[,
25]),
                              sd.avgR = as.numeric(MCMC_sumtemp[,26]),
                              avgR2.5 = as.numeric(MCMC_sumtemp[,27]),
                              avgR97.5 = as.numeric(MCMC_sumtemp[,28]),
                              mean.avgM = as.numeric(MCMC_sumtemp[,
29]),
                              sd.avgM = as.numeric(MCMC_sumtemp[,30]),
                              avgM2.5 = as.numeric(MCMC_sumtemp[,31]),
                              avgM97.5 = as.numeric(MCMC_sumtemp[,32]))

save(MCMC_summary, file = paste0("MCMC_summary_singlerun.rda"))


# plot single run with original pmult data (from Avis)
g1a <- ggplot() +
  stat_smooth(data = MCMC_summary, aes(avgbrood, mean.est.pmult),
color = "red", method = "loess") +
  geom_point(data = MCMC_summary, aes(avgbrood, mean.est.pmult), color
= "red", alpha = 0.2, size = 2.5) +
  geom_point(data = dat, aes(avgbrood, pmult)) +
  labs(x="Litter size", y="Probability of multiple paternity") +
  lims(y = c(0,1)) +
  scale_x_continuous(breaks = c(seq(2,10, by = 1)) , limits = c(2,10))
+
  theme_bw(base_size = 16)
ggsave("p_singlerun.eps", plot = g1a, device = cairo_ps, width = 11,
height = 7.5, dpi = 320)



################ CALCULATE RESIDUALS ##################
MCMC_resids <- MCMC_summary
loess_pmult <- loess(mean.est.pmult ~ avgbrood, data = MCMC_resids)
MCMC_resids$pred_pmult <- predict(loess_pmult, newdata =
MCMC_resids$avgbrood)
# lower limit
loess_pmult_lcl <- loess(est.pmult2.5 ~ avgbrood, data = MCMC_resids)
MCMC_resids$pred_pmult_lcl <- predict(loess_pmult_lcl, newdata =
MCMC_resids$avgbrood)
# upper limit
loess_pmult_ucl <- loess(est.pmult97.5 ~ avgbrood, data = MCMC_resids)
MCMC_resids$pred_pmult_ucl <- predict(loess_pmult_ucl, newdata =
MCMC_resids$avgbrood)
# residual calculations
MCMC_resids$resid_pmult <- MCMC_resids$pred_pmult - MCMC_resids$pmult
MCMC_resids$resid_pmult_lcl <- MCMC_resids$pred_pmult_lcl -
```

```
MCMC_resids$pmult
MCMC_resids$resid_pmult_ucl <- MCMC_resids$pred_pmult_ucl -
MCMC_resids$pmult

# save dataframe with all residual calculations
save(MCMC_resids, file = "MCMC_resids.rda")
write.csv(MCMC_resids, file = "MCMC_resids.csv")

# plot pred_pmult with lower and upper limits
g1b <- ggplot() +
  geom_point(data = dat, aes(avgbrood, pmult), alpha = 0.5, size =
2.5) +
  stat_smooth(data = dat, aes(avgbrood, pmult), se = FALSE, color =
"black", method = "lm") +
  stat_smooth(data = MCMC_resids, aes(avgbrood, pred_pmult), color =
"red", se = FALSE, method = "loess") +
  geom_point(data = MCMC_resids, aes(avgbrood, mean.est.pmult), color
= "red", alpha = 0.5, size = 2.5) +
  stat_smooth(data = MCMC_resids, aes(avgbrood, pred_pmult_ucl), color
= "blue", se = FALSE, size = 0.5, method = "loess") +
  stat_smooth(data = MCMC_resids, aes(avgbrood, pred_pmult_lcl), color
= "blue", se = FALSE, size = 0.5, method = "loess") +
  labs(x="Litter size", y="Probability of multiple paternity") +
  lims(y = c(0,1)) +
  scale_x_continuous(breaks = c(seq(2,10, by = 1)) , limits = c(2,10))
+
  theme_bw(base_size = 16)
ggsave("p_CL.eps", plot = g1b, device = cairo_ps, width = 11, height =
7.5, dpi = 320)


################ CALCULATE EFFECT SIZES ##################
mammals_1 <- read.csv("paternity_mammals.csv", header = TRUE)
mammals_1 <- mammals_1[!is.na(mammals_1$avgsire),]

mammals_singlerun <- merge(mammals_1[,c(1:3)], MCMC_resids, by =
c("species", "avgbrood"))
# order by brood size
mammals_singlerun <-
mammals_singlerun[order(mammals_singlerun$avgbrood,
mammals_singlerun$pmult, mammals_singlerun$nbrood),]

### variance p(1-p)/n for fixed p assumption
mammals_singlerun$vi <- mammals_singlerun$pmult*(1-
mammals_singlerun$pmult)/mammals_singlerun$nbrood

library(metafor)
paternity.meta.bayes <- rma(resid_pmult, vi, data = mammals_singlerun)
paternity.meta.bayes
pdf(file = paste0("forest_bayes_singlerun-w-labels.pdf"), width =
```

```
11.5, height = 13)
print(
  forest(paternity.meta.bayes, slab = mammals_singlerun$species,
         order = order(mammals_singlerun$avgbrood), cex = 1,
         xlim = c(-2.5,2),
         xlab = expression(paste(p[B], " - p"))),
  quote = FALSE,
  text(-2.5, 62, "Species",  pos=4),
  text(2, 62, expression(paste(p[B], " - p [95% CI]")), pos=2)
)
dev.off()

# k as moderator
paternity.meta.k <- rma(resid_pmult, vi, mods = ~ avgbrood, data =
mammals_singlerun)
paternity.meta.k
pdf(file = paste0("forest_bayes_singlerun_k-mod.pdf"), width = 15,
height = 12)
print(
  forest(paternity.meta.k, slab = mammals_singlerun$species,
         order = order(mammals_singlerun$avgbrood), cex = 1)
)
dev.off()



################ E(M) AND E(S) VS K PLOT ##################
## Plot estimated M from data using Bayesian p overlaid on observed
avg M vs obs. littersize
loess_R <- loess(mean.avgR ~ avgbrood, data = MCMC_resids)
MCMC_resids$pred_R <- predict(loess_R, newdata = MCMC_resids$avgbrood)
# lower limit
loess_R_lcl <- loess(avgR2.5 ~ avgbrood, data = MCMC_resids)
MCMC_resids$pred_R_lcl <- predict(loess_R_lcl, newdata =
MCMC_resids$avgbrood)
# upper limit
loess_R_ucl <- loess(avgR97.5 ~ avgbrood, data = MCMC_resids)
MCMC_resids$pred_R_ucl <- predict(loess_R_ucl, newdata =
MCMC_resids$avgbrood)

g2 <- ggplot() +
  stat_smooth(data = MCMC_resids, aes(avgbrood, pred_R), color =
"red", se = FALSE, method = "loess") +
  geom_point(data = MCMC_resids, aes(avgbrood, mean.avgR), color =
"red", alpha = 0.5, size = 2.5) +
  stat_smooth(data = MCMC_resids, aes(avgbrood, pred_R_ucl), color =
"blue", se = FALSE, size = 0.5, method = "loess") +
  stat_smooth(data = MCMC_resids, aes(avgbrood, pred_R_lcl), color =
"blue", se = FALSE, size = 0.5, method = "loess") +
  stat_smooth(data = mammals_1, aes(avgbrood, avgsire), color =
```

```
"black", method = "lm", se = FALSE) +
  geom_point(data = mammals_1, aes(avgbrood, avgsire), alpha = 0.5,
size = 2.5) +
  labs(x="Litter size", y="Number of sires") +
  scale_x_continuous(breaks = c(seq(1,10, by = 1)) , limits = c(2,10))
+
  theme_bw(base_size = 16)
ggsave("R_CL.eps", plot = g2, device = cairo_ps, width = 11, height =
7.5, dpi = 320)


## Plot of #mates/sires vs brood size with Bayesian estimates

# Combinatorics avgmates calculated using pred_pmult and true k
mammals_1$avgmate <- exp(log(1-mammals_1$pmult)/(1-
mammals_1$avgbrood))

# Used the Avise data avgbrood and avgsire to obtain avgR = k*q/(1-(1-
q)^k) and avgM = r/q
# calculate predicted M
loess_M <- loess(mean.avgM ~ avgbrood, data = MCMC_resids)
MCMC_resids$pred_M <- predict(loess_M, newdata = MCMC_resids$avgbrood)
# plot
g3 <- ggplot() +
  # avgR in red is calculated from mean of ZTB dist, since S ~ ZTB
  stat_smooth(data = MCMC_resids, aes(avgbrood, pred_R, linetype =
"Estimated # sires using\ndata K and Bayesian q"), color = "red",
method = "loess", se = TRUE) +
  # points from which the above line was generated:
  #geom_point(data = MCMC_resids, aes(avgbrood, mean.avgR), color =
"red", alpha = 0.2) +
  # True number of average sires from the Avise data
  geom_point(data = mammals_1, aes(avgbrood, avgsire, color =
"Observed number of sires"), alpha = 0.5, size = 2.5) +
  # Avise used avgsires as estimated mates (i.e. he does not
distinguish b/w mates and sires?)
  stat_smooth(data = mammals_1, aes(avgbrood, avgsire, linetype =
"Observed number of sires"), color = "black", method = "lm", se =
FALSE) +
  # avgM in blue is calculated from mean of NB dist knowing S and q,
since M ~ NB
  stat_smooth(data = MCMC_resids, aes(avgbrood, pred_M, linetype =
"Estimated # mates using\ndata S and Bayesian q"), color = "blue",
method = "lm", se = TRUE) +
  # points from which the above line was generated:
  #geom_point(data = MCMC_resids, aes(avgbrood, mean.avgM), color =
"blue", alpha = 0.2) +
  # avgmates calculated using true pmult and true k
  stat_smooth(data = mammals_1[!is.infinite(mammals_1$avgmate),],
aes(avgbrood, avgmate, linetype = "Estimated # mates
```

```
using\ncombinatorial formula"),
              color = "green3", method = "lm", se = TRUE) +
  geom_point(data = mammals_1[!is.infinite(mammals_1$avgmate),],
aes(avgbrood, avgmate, color = "Estimated # mates using\ncombinatorial
formula"), alpha = 0.5, size = 2.5) +
  labs(x="Litter size", y="Number of mates/sires") +
  scale_x_continuous(breaks = c(seq(1,10, by = 1)) , limits = c(1,10))
+
  scale_y_continuous(limits = c(0,8)) +
  scale_colour_manual(name = "", values = c("Observed number of sires"
= "black",
                                            "Estimated # mates
using\ncombinatorial formula" = "green3")) +
  scale_linetype_manual(values = c("Estimated # mates using\ndata S
and Bayesian q" = 1, # blue line
                                   "Estimated # mates
using\ncombinatorial formula" = 1, # dark green line
                                   "Estimated # sires using\ndata K
and Bayesian q" = 1, # red line
                                   "Observed number of sires" = 1), #
black line
                         name = "",
                         guide = guide_legend(override.aes = list(color
= c("blue", "green3", "red", "black"),
                                                                  fill
= c(NA,NA,NA,NA)))) +
  theme_bw(base_size = 16) + theme(legend.key.height=unit(15, "mm"))
ggsave("Avise_MS-k.eps", plot = g3, device = cairo_ps, width = 12,
height = 7.5, dpi = 320)


# Used the generated data to create nsires and calculate estM =
nsires/q
g4 <- ggplot() +
  # NB estimated mates using different values of q
  stat_function(fun = function(x) x/(1-0.2^x), aes(x = x, linetype =
"Expected # mates from NB distribution"), data = data.frame(x =
c(2,10)), color = "magenta", size = 1.2) +
  stat_function(fun = function(x) x/(1-0.3^x), aes(x = x, linetype =
"Expected # mates from NB distribution"), data = data.frame(x =
c(2,10)), color = "magenta", size = 1.2) +
  stat_function(fun = function(x) x/(1-0.4^x), aes(x = x, linetype =
"Expected # mates from NB distribution"), data = data.frame(x =
c(2,10)), color = "magenta", size = 1.2) +
  stat_function(fun = function(x) x/(1-0.5^x), aes(x = x, linetype =
"Expected # mates from NB distribution"), data = data.frame(x =
c(2,10)), color = "magenta", size = 1.2) +
  # # nsires in red is generated from Poission dist with lambda =
avgsire in Avise data
  # stat_smooth(data = MCMC_resids, aes(avgbrood, mean.nsires,
```

```
linetype = "Generated # sires using Poisson(avgsire)"), color = "red",
method = "lm", se = TRUE) +
  # # points from which the above line was generated:
  # geom_point(data = MCMC_resids, aes(avgbrood, mean.nsires, color =
"Generated # sires using Poisson(avgsire)"), alpha = 0.5) +
  # True number of average sires from the Avise data
  geom_point(data = mammals_1, aes(avgbrood, avgsire, color =
"Observed number of sires"), alpha = 0.5, size = 2.5) +
  # Avise used avgsires as estimated mates (i.e. he does not
distinguish b/w mates and sires?)
  stat_smooth(data = mammals_1, aes(avgbrood, avgsire, linetype =
"Observed number of sires"), color = "black", method = "lm", se =
FALSE) +
  # estM in blue is calculated from mean of NB dist knowing generated
S and q, since M ~ NB
  stat_smooth(data = MCMC_resids, aes(avgbrood, mean.estM, linetype =
"Bayesian MCMC estimated # mates"), color = "blue", method = "lm", se
= FALSE) +
  # points from which the above line was generated:
  geom_point(data = MCMC_resids, aes(avgbrood, mean.estM, color =
"Bayesian MCMC estimated # mates"), alpha = 0.5, size = 2.5) +
  # avgmates calculated using true pmult and true k
  stat_smooth(data = mammals_1[!is.infinite(mammals_1$avgmate),],
aes(avgbrood, avgmate, linetype = "Estimated # mates
using\ncombinatorial formula"),
              color = "green3", method = "lm", se = FALSE) +
  geom_point(data = mammals_1[!is.infinite(mammals_1$avgmate),],
aes(avgbrood, avgmate, color = "Estimated # mates using\ncombinatorial
formula"), alpha = 0.5, size = 2.5) +
  labs(x="Litter size", y="Number of mates/sires") +
  scale_x_continuous(breaks = c(seq(1,10, by = 1)) , limits = c(2,10))
+
  scale_y_continuous(breaks = c(seq(0,10, by = 2)) ,limits = c(0,10))
+
  scale_colour_manual(name = "", values = c("Bayesian MCMC estimated #
mates" = "blue",
                                            "Estimated # mates
using\ncombinatorial formula" = "green3",
                                            #"Generated # sires using
Poisson(avgsire)" = "red",
                                            "Observed number of sires"
= "black")) +
  scale_linetype_manual(values = c("Bayesian MCMC estimated # mates" =
1, # blue line,
                                   "Estimated # mates
using\ncombinatorial formula" = 1, # green3 line
                                   "Expected # mates from NB
distribution" = 1, # magenta line
                                   #"Generated # sires using
Poisson(avgsire)" = 1, # red line
```

```
                                    "Observed number of sires" = 1), #
black line
                          name = "",
                          guide = guide_legend(override.aes = list(color
= c("blue","green3",

"magenta",

#"red",

"black"),
                                                                    fill
= c(NA,NA,

NA,

#NA,

NA)))) +
  theme_bw(base_size = 16) + theme(legend.key.height=unit(10, "mm"))
ggsave("Bayes_gen_MS-k.eps", plot = g4, device = cairo_ps, width = 12,
height = 7.5, dpi = 320)
```