Supporting Information

Separating Measurement Error and Signal in Environmental Data: Use of Replicates to Address Uncertainty

Marschall Furman[1], Kent W. Thomas[2], Barbara Jane George[2*]

[1]Oak Ridge Institute for Science and Education (ORISE) Research Participant at U.S. EPA, Office of Research and Development, Center for Public Health and Environmental Assessment, Research Triangle Park, North Carolina 27711, United States

[2]Center for Public Health and Environmental Assessment, Office of Research and Development, U.S.
EPA, Research Triangle Park, North Carolina 27711, United States

*Address correspondence to B.J. George, CPHEA/ORD/U.S. EPA, 109 T.W. Alexander Dr., Research
Triangle Park, NC 27711 USA. Telephone: (919) 541-4551. E-mail: george.bj@epa.gov

Table of Contents

Note: Supplemental figures have been reorded since initial submission. Descriptions in comments are accurate but many figure numbers may no longer apply.

Simulation Code - R

01_generate_simulation_data.R

- define functions and settings for simulating data
- create lists with data frames for 1-2 and 2-3
- calculate Naive and Hybrid estimates
- outputs: `ME_simulation_data.Rdata`, `ME_simulation_data_J=3.Rdata`

```
## GOAL: generate data for all simulation settings, obtain Naive/Hybrid
estimates.

# functions ------------------------------------------------------------

## function to generate data all at once
generate_data <- function(pgrid, nsims, myseed=1994){

  set.seed(myseed)

  # number of replicates
  Jcur=2

  # critical value for CI
  nom=0.05

  # number of settings to loop over
  nsets <- nrow(pgrid)

  # data storage
  all_data <- vector("list", nsets)
  naive_ci <- vector("list", nsets)
  naive_est <- vector('list', nsets)
  hybrid_ci <- vector('list', nsets)
  hybrid_est <- vector('list', nsets)

  for(gr in 1:nsets){

    # fix parameter settings
```

```
ncur <- pgrid$n[gr]
sig2_x_cur <- pgrid$sig2_x[gr]
sig2_e_cur <- pgrid$sig2_e[gr]
pctJcur <- pgrid$pctJ[gr]

# nrep = # of replicates per sample, ncur2 = # of replicated samples
nrep <- c(rep(Jcur, round((1-pctJcur)*ncur)),
          rep(Jcur-1, round(pctJcur*ncur)))[1:ncur]
ncur2 <- sum(nrep==Jcur)

# data storage
cur_data <- matrix(NA, nc=nsims, nr=ncur+ncur2)
naive_ci_cur <- matrix(NA, nr=nsims, nc=2)
hybrid_ci_cur <- matrix(NA, nr=nsims, nc=2)
naive_est_cur <- matrix(NA, nr=nsims, nc=2)
hybrid_est_cur <- matrix(NA, nr=nsims, nc=2)

for(i in 1:nsims){

  ## true, unobserved signal data
  x <- rnorm(ncur, mean=mu_x, sd=sqrt(sig2_x_cur))

  # repeated observations (same signal)
  x2 <- matrix(NA, nr=2, ncol=ncur)
  x2[1,] <- x
  x2[2,1:ncur2] <- x[1:ncur2]

  ## measurement errors for 1st replicate
  e <- rnorm(ncur, mean=0, sd=sqrt(sig2_e_cur))

  ## measurement errors for 2nd replicate
  e2 <- matrix(NA, nr=2, ncol=ncur)
  e2[1,] <- e
  e2[2,1:ncur2] <- rnorm(ncur2, 0, sqrt(sig2_e_cur))

  ## observed data
  y2 <- x2 + e2

  ## naive analysis
  Y_naive <- y2[1,]
  Ybar_naive <- mean(Y_naive, na.rm=T)
  nnaive <- length(Y_naive)
  var_naive <- var(Y_naive, na.rm=T)

  ci_Ybar_naive <- Ybar_naive + c(-1,1) *
    qt(1-nom/2, df=ncur-1) * sd(Y_naive)/sqrt(nnaive)

  naive_est_cur[i,] <- c(Ybar_naive, var_naive)
  naive_ci_cur[i,] <- ci_Ybar_naive
```

```
      ## hybrid analysis
      Y_hybrid <- y2[!is.na(y2)]
      Ybar_hybrid <- mean(Y_hybrid, na.rm=T)
      var_hybrid <- var(Y_hybrid, na.rm=T)
      nhybrid <- length(Y_hybrid)

      ci_Ybar_hybrid <- Ybar_hybrid +
        c(-1,1)*qt(1-nom/2, df=nhybrid-1)*
        sqrt(var(Y_hybrid)/nhybrid)

      hybrid_est_cur[i,] <- c(Ybar_hybrid, var_hybrid)
      hybrid_ci_cur[i,] <- ci_Ybar_hybrid

      ## remove missing values
      cur_data[,i] <- c(y2[!is.na(y2)])

    }
    # set up datasets to feed into REML model
    grp2 <- factor(rep(1:ncur, times=nrep))
    alldat <- sapply(1:nsims, function(a) data.frame(Y=cur_data[,a],grp2),
                     simplify=F)

    colnames(naive_ci_cur) <- c('mean_ci1_naive', 'mean_ci2_naive')
    colnames(hybrid_ci_cur) <- c('mean_ci1_hybrid', 'mean_ci2_hybrid')
    colnames(naive_est_cur) <- c('mean_est_naive', 'var_est_naive')
    colnames(hybrid_est_cur) <- c('mean_est_hybrid', 'var_est_hybrid')

    all_data[[gr]] <- alldat
    naive_ci[[gr]] <- naive_ci_cur
    hybrid_ci[[gr]] <- hybrid_ci_cur

    naive_est[[gr]] <- naive_est_cur
    hybrid_est[[gr]] <- hybrid_est_cur
  }
  return(list('data'=all_data,'naive_ci'=naive_ci,'naive_est'=naive_est,
              'hybrid_ci' = hybrid_ci, 'hybrid_est'=hybrid_est))

}

## same function, but for 2-3 replicates
generate_dataJ3 <- function(pgrid, nsims, myseed=1994){

  set.seed(myseed)

  # number of replicates
  Jcur <- 3

  # number of settings to loop over
```

```r
nsets <- nrow(pgrid)

# critical value for CI
nom <- 0.05

# data storage
all_data <- vector("list", nrow(pgrid))
naive_ci <- vector("list", nsets)
naive_est <- vector('list', nsets)
hybrid_ci <- vector('list', nsets)
hybrid_est <- vector('list', nsets)

for(gr in 1:nrow(pgrid)){

  # fix parameter settings
  ncur <- pgrid$n[gr]
  sig2_x_cur <- pgrid$sig2_x[gr]
  sig2_e_cur <- pgrid$sig2_e[gr]
  pctJcur <- pgrid$pctJ[gr]

  # nrep = # of replicates per sample, ncur2 = # of replicated samples
  nrep <- c(rep(Jcur, round((1-pctJcur)*ncur)),
            rep(Jcur-1, round(pctJcur*ncur)))[1:ncur]
  ncur2 <- sum(nrep == Jcur)

  # data storage
  cur_data <- matrix(NA, nc=nsims, nr=2*ncur+ncur2)
  naive_ci_cur <- matrix(NA, nr=nsims, nc=2)
  hybrid_ci_cur <- matrix(NA, nr=nsims, nc=2)
  naive_est_cur <- matrix(NA, nr=nsims, nc=2)
  hybrid_est_cur <- matrix(NA, nr=nsims, nc=2)

  for(i in 1:nsims){

    ## true, unobserved signal data
    x <- rnorm(ncur, mean=mu_x, sd=sqrt(sig2_x_cur))

    # repeated observations (same signal)
    x2 <- matrix(NA, nr=3, ncol=ncur)
    x2[1,] <- x
    x2[2,] <- x
    x2[3,1:ncur2] <- x[1:ncur2]

    ## measurement errors for 1st replicate
    e <- rnorm(ncur, mean=0, sd=sqrt(sig2_e_cur))
    ee <- rnorm(ncur, mean=0, sd=sqrt(sig2_e_cur))

    ## measurement errors for 2nd replicate
    e2 <- matrix(NA, nr=3, ncol=ncur)
```

```
      e2[1,] <- e
      e2[2,] <- ee
      e2[3,1:ncur2] <- rnorm(ncur2, 0, sqrt(sig2_e_cur))

      ## observed data
      y2 <- x2 + e2

      ## naive analysis
      Y_naive <- y2[1,]
      Ybar_naive <- mean(Y_naive, na.rm=T)
      nnaive <- length(Y_naive)
      var_naive <- var(Y_naive, na.rm=T)

      ci_Ybar_naive <- Ybar_naive + c(-1,1) *
        qt(1-nom/2, df=ncur-1) *
        sd(Y_naive)/sqrt(nnaive)

      naive_est_cur[i,] <- c(Ybar_naive, var_naive)
      naive_ci_cur[i,] <- ci_Ybar_naive

      ## hybrid analysis
      Y_hybrid <- y2[!is.na(y2)]
      Ybar_hybrid <- mean(Y_hybrid, na.rm=T)
      var_hybrid <- var(Y_hybrid, na.rm=T)
      nhybrid <- length(Y_hybrid)

      ci_Ybar_hybrid <- Ybar_hybrid +
        c(-1,1)*qt(1-nom/2, df=nhybrid-1)*
        sqrt(var(Y_hybrid)/nhybrid)

      hybrid_est_cur[i,] <- c(Ybar_hybrid, var_hybrid)
      hybrid_ci_cur[i,] <- ci_Ybar_hybrid

      cur_data[,i] <- c(y2[!is.na(y2)])

    }
    # set up datasets to feed into REML model
    grp2 <- factor(rep(1:ncur, times=nrep))
    alldat <- sapply(1:nsims, function(x) data.frame(Y=cur_data[,x],grp2),
simplify=F)

    ## adjust column names
    colnames(naive_ci_cur) <- c('mean_ci1_naive', 'mean_ci2_naive')
    colnames(hybrid_ci_cur) <- c('mean_ci1_hybrid', 'mean_ci2_hybrid')
    colnames(naive_est_cur) <- c('mean_est_naive', 'var_est_naive')
    colnames(hybrid_est_cur) <- c('mean_est_hybrid', 'var_est_hybrid')

    ## save current dataset & estimates
    all_data[[gr]] <- alldat
```

```
    naive_ci[[gr]] <- naive_ci_cur
    hybrid_ci[[gr]] <- hybrid_ci_cur

    naive_est[[gr]] <- naive_est_cur
    hybrid_est[[gr]] <- hybrid_est_cur
  }

  ## return data and naive + hybrid estimates
  return(list('data' = all_data, 'naive_ci' = naive_ci, 'naive_est' =
naive_est,
              'hybrid_ci' = hybrid_ci, 'hybrid_est' = hybrid_est))
}


# define settings ----------------------------------------------------------

nsims <- 100 # number of simulations

mu_x <- 10 # population mean
sig2_x <- c(3,9,15) # signal variance levels
sig2_e <- c(0.25,1,2.25,4,6.25) # ME variance levels

## create dataframe of all 3*10*5*3 = 450 settings
pgrid <- expand.grid(n = c(20, 50,100), pctJ = seq(0,0.9,by=0.1),
                     mu_x = mu_x, sig2_x = sig2_x, sig2_e = sig2_e)

nsets <- nrow(pgrid)

# generate data ------------------------------------------------------------

## 1-2 replicates (J=2)
time1_data <- Sys.time()
dat <- generate_data(pgrid, nsims = nsims)
time2_data <- Sys.time()

## print runtime
difftime(time2_data, time1_data, units = 'mins')

## save output
saveRDS(dat, file = 'ME_simulation_data.Rdata')

## 2-3 replicates (J=3)
time3_data <- Sys.time()
dat_J3 <- generate_dataJ3(pgrid, nsims=nsims)
time4_data <- Sys.time()

## print runtime
difftime(time4_data, time3_data, units = 'mins')
```

```
## save output
saveRDS(dat_J3, file = 'ME_simulation_data_J=3.Rdata')
```

02_analyze_simulation_data.R

- fit random effects models using `lme4` package
- setup paralellization for faster processing
- outputs: `ME_simulation_output.Rdata`, `ME_simulation_output_J=3.Rdata`

```
library(parallel) # for speeding up analysis through parallelization

## GOAL: use parallel processing to fit MEM models, extract coefs and CIs

# functions ------------------------------------------------------------------


## analyze all data at once - not parallel!
run_MEM_serial <- function(pgrid, Ydata, nsims){

  ## number of settings
  nsets <- nrow(pgrid)

  ## data storage
  mem_list <- vector("list", nsets)
  mem_est <- matrix(NA, nrow=nsims, nc=3)
  mem_ci <- matrix(NA, nrow=nsims, nc=2)
  colnames(mem_est) <- c('mean_est','sig2_x_est','sig2_e_est')
  colnames(mem_ci) <- c('mean_ci1_mem', 'mean_ci2_mem')

  ## loop over simulation settings
  for(gr in 1:nsets){

    ## loop over each dataset
    for(i in 1:nsims){
      alldat <- Ydata[[gr]][[i]]

      ## fit random effects model
      fit.mem <- lmer(Y ~ (1 | grp2), data = alldat, REML = TRUE)

      ## extract model coefficients
      mem_est[i,] <- c(fit.mem@beta, as.data.frame(VarCorr(fit.mem))$vcov)

      ## compute Kenward-Roger confidence interval
      kr_ci <- contest(fit.mem, L=1, joint=FALSE, confint = TRUE,
                       ddf = "Kenward-Roger", level = 0.95)
      mem_ci[i,] <- c(kr_ci$lower, kr_ci$upper)
    }

    mem_list[[gr]] <- list('est' = mem_est, 'ci' = mem_ci)
```

```
    print(gr)
  }


  return(mem_list)

}

## analyze data, parallellized by sim settings
run_MEM_parallel <- function(index, Ydata, nsims){

  ## create objects for storing results
  mem_est <- matrix(NA, nrow=nsims, nc=3)
  mem_ci <- matrix(NA, nrow=nsims, nc=2)
  colnames(mem_est) <- c('mean_est','sig2_x_est','sig2_e_est')
  colnames(mem_ci) <- c('mean_ci1_mem', 'mean_ci2_mem')

  mem_warn <- rep(NA, nsims)

  ## index data for one setting
  Ydata_indexed <- Ydata[[index]]
  print(paste0("Starting setting ", index," of ", length(Ydata)))

  ## loop over each dataset
  for(i in 1:nsims){

    ## index single dataset
    alldat <- Ydata_indexed[[i]]

    ## fit random effects model
    fit.mem <- lmer(Y ~ (1 | grp2), data=alldat, REML=TRUE)

    ## check if model converged
    if(!is.null(fit.mem@optinfo$conv$lme4$messages)){
      mem_warn[i] <- fit.mem@optinfo$conv$lme4$messages

    }

    ## extract model coefficients
    mem_est[i,] <- c(fit.mem@beta, as.data.frame(VarCorr(fit.mem))$vcov)

    ## compute 95% CI, with a Kenward-Roger method
    kr_ci <- contest(fit.mem, L=1, joint=FALSE, confint = TRUE,
                     ddf = "Kenward-Roger", level = 0.95)
    mem_ci[i,] <- c(kr_ci$lower, kr_ci$upper)

  }
```

```
  return(list('index' = index, 'est' = mem_est,
              'ci' = mem_ci, 'warn' = mem_warn))

}


# analyze J=2 data ----------------------------------------------------


## read in generated data, J=2
dat <- readRDS('ME_simulation_data.Rdata')

## initiate cluster with 4 cores
cl <- makeCluster(4, outfile = 'parallel_log_J=2.txt')

## pass packages to each core
clusterEvalQ(cl,
             {
               library(lme4)
               library(lmerTest)
               })

## run full parallelized analysis, J=2
time1_res <- Sys.time()
res_split <- parallel::parLapply(cl = cl, X = 1:nsets,
                                 fun = run_MEM_parallel,
                                 Ydata = dat$data, nsims = nsims)
time2_res <- Sys.time()

## print runtime
difftime(time2_res, time1_res, units = 'mins')

## save J=2 output
saveRDS(res_split, file = 'ME_simulation_output.Rdata')

## stop running J=2 cluster
stopCluster(cl)



# analyze J=3 data ----------------------------------------------------


## read in generated data, J=3
dat_J3 <- readRDS('ME_simulation_data_J=3.Rdata')


## initiate cluster with 4 cores
```

```
cl_J3 <- makeCluster(4, outfile='parallel_log_J=3.txt')

## pass packages to each core
clusterEvalQ(cl_J3,
             {
               library(lme4)
               library(lmerTest)
             })

## run full parallelized analysis, J=3
time3_res <- Sys.time()
res_split_J3 <- parallel::parLapply(cl = cl_J3, X = 1:nsets,
                                    fun = run_MEM_parallel,
                                    Ydata = dat_J3$data, nsims = nsims)
time4_res <- Sys.time()

## print runtime
difftime(time4_res, time3_res, units = 'mins')

## save J=3 output
saveRDS(res_split_J3, file = 'ME_simulation_output_J=3.Rdata')

## stop running J=3 cluster
stopCluster(cl_J3)
```

03_process_simulation_results.R

- compute summary stats (bias, coverage, etc.) of all 3 approaches
- arrange into large data frames for plotting purposes
- outputs: `ME_processed_output.Rdata`, `ME_figure_data_summary.Rdata`, `ME_processed_output_J=3.Rdata`, `ME_figure_data_summary_J=3.Rdata`

```
library(tidyverse)

## GOAL: combines output from scripts 1 and 2, stores in full (450,000 rows)
##       and summary (450 rows) formats
# aggregate J=2 results ------------------------------------------------


## read in J=2 simulation data and results
dat <- readRDS('ME_simulation_data.Rdata')
res_split <- readRDS('ME_simulation_output.Rdata')

## combine Naive and Hybrid estimates with MEM model output
for(i in 1:nsets){

  res_split[[i]]$est_naive <- dat$naive_est[[i]]
```

```
  res_split[[i]]$est_hybrid <- dat$hybrid_est[[i]]
  res_split[[i]]$ci_naive <- dat$naive_ci[[i]]
  res_split[[i]]$ci_hybrid <- dat$hybrid_ci[[i]]

}

## extract estimates and CIs from each simulation setting
df_out <- data.frame()

for(k in 1:nsets){

  df_cur <- data.frame(
    setting_id = rep(res_split[[k]]$index, nsims),
    sim = 1:nsims,
    res_split[[k]]$est,
    res_split[[k]]$ci,
    res_split[[k]]$est_naive,
    res_split[[k]]$ci_naive,
    res_split[[k]]$est_hybrid,
    res_split[[k]]$ci_hybrid,
    singular = !is.na(res_split[[k]]$warn)
  )

  df_out <- rbind(df_out, df_cur)
  print(k)
}

## combine setting variables and results
pgrid <- pgrid %>% mutate(setting_id = row_number())
df_out <- left_join(pgrid, df_out, by = c('setting_id'))

## save full output (rows = 450 settings x nsims)
res_long <- df_out %>%
  arrange(sig2_e, sig2_x, pctJ, n, sim) %>%
  group_by(sig2_e, sig2_x, pctJ, n) %>%
  mutate(pct_warn = mean(singular == T)) %>%
  ungroup()

saveRDS(res_long, 'ME_processed_output.Rdata')

## summarize within each simulation setting (rows = 450)
res_long_summary <- res_long %>%
  ## add variables comparing naive and MEM variance estimates
  mutate(
         naive_smaller_signal = var_est_naive < sig2_x_est,
         naive_smaller_total = var_est_naive < (sig2_x_est + sig2_e_est),
         mean_ci1_signal = mean_est - qt(0.975, df = n-1) *
sqrt(sig2_x_est/n),
         mean_ci2_signal = mean_est + qt(0.975, df = n-1) *
```

```
sqrt(sig2_x_est/n)) %>%
  ## drop simulations that could not fit MEM models
  filter(singular == F) %>%
  group_by(sig2_e, sig2_x, pctJ, n) %>%
  summarise(
    ## compute averages of means and variances
    across(c(mean_est, sig2_x_est, sig2_e_est,
             var_est_naive, var_est_hybrid, mean_est,
             mean_est_naive, mean_est_hybrid, pct_warn), .fns = mean),
    ## compute confidence interval coverage (% of sims where true value
inside)
    covg_mem = mean(mean_ci1_mem <= mu_x & mu_x <= mean_ci2_mem),
    covg_naive = mean(mean_ci1_naive <= mu_x & mu_x <= mean_ci2_naive),
    covg_hybrid = mean(mean_ci1_hybrid <= mu_x & mu_x <= mean_ci2_hybrid),
    covg_signal = mean(mean_ci1_signal <= mu_x & mu_x <= mean_ci2_signal),
    ## compute confidence interval widths
    ci_width_hybrid = mean(mean_ci2_hybrid - mean_ci1_hybrid),
    ci_width_naive = mean(mean_ci2_naive- mean_ci1_naive),
    ci_width_mem = mean(mean_ci2_mem - mean_ci1_mem),
    ci_width_signal = mean(mean_ci2_signal - mean_ci1_signal)
  ) %>%
  ungroup() %>%
  ## extra variables for plotting
  mutate(sig2e_str = 'True ME Variance', sig2x_str = 'True Signal Variance',
         n_str = 'Sample Size', pct_rep = 100 - 100*pctJ)

## save summary data
saveRDS(res_long_summary, 'ME_figure_data_summary.Rdata')


# aggregate J=3 results ---------------------------------------------------

## read in J=3 simulation data and results
dat_J3 <- readRDS('ME_simulation_data_J=3.Rdata')
res_split_J3 <- readRDS('ME_simulation_output_J=3.Rdata')

## combine Naive and Hybrid estimates with MEM model output
for(i in 1:nsets){

  res_split_J3[[i]]$est_naive <- dat_J3$naive_est[[i]]
  res_split_J3[[i]]$est_hybrid <- dat_J3$hybrid_est[[i]]
  res_split_J3[[i]]$ci_naive <- dat_J3$naive_ci[[i]]
  res_split_J3[[i]]$ci_hybrid <- dat_J3$hybrid_ci[[i]]

}

## extract estimates and CIs from each simulation setting
df_out_J3 <- data.frame()
```

```r
for(k in 1:nsets){

  df_cur_J3 <- data.frame(
    setting_id = rep(res_split_J3[[k]]$index, nsims),
    sim = 1:nsims,
    res_split_J3[[k]]$est,
    res_split_J3[[k]]$ci,
    res_split_J3[[k]]$est_naive,
    res_split_J3[[k]]$ci_naive,
    res_split_J3[[k]]$est_hybrid,
    res_split_J3[[k]]$ci_hybrid,
    singular = !is.na(res_split_J3[[k]]$warn)
  )

  df_out_J3 <- rbind(df_out_J3, df_cur_J3)
  print(k)
}

## combine settings and results
pgrid <- pgrid %>% mutate(setting_id = row_number())
df_out_J3 <- left_join(pgrid, df_out_J3, by = c('setting_id'))

## save full output (rows = 450 settings x nsims)
res_long_J3 <- df_out_J3 %>%
  arrange(sig2_e, sig2_x, pctJ, n, sim) %>%
  group_by(sig2_e, sig2_x, pctJ, n) %>%
  mutate(pct_warn = mean(singular == T)) %>%
  ungroup()

saveRDS(res_long_J3, 'ME_processed_output_J=3.Rdata')

## summarize within each simulation setting (rows = 450)
res_long_summary_J3 <- res_long_J3 %>%
  ## add variables comparing naive and MEM variance estimates
  mutate(
    naive_smaller_signal = var_est_naive < sig2_x_est,
    naive_smaller_total = var_est_naive < (sig2_x_est + sig2_e_est),
    mean_ci1_signal = mean_est - qt(0.975, df = n-1) * sqrt(sig2_x_est/n),
    mean_ci2_signal = mean_est + qt(0.975, df = n-1) * sqrt(sig2_x_est/n))
%>%
  ## drop simulations that could not fit MEM models
  filter(singular == F) %>%
  group_by(sig2_e, sig2_x, pctJ, n) %>%
  summarise(
    ## compute averages of means and variances
    across(c(mean_est, sig2_x_est, sig2_e_est,
             var_est_naive, var_est_hybrid, mean_est,
             mean_est_naive, mean_est_hybrid, pct_warn), .fns = mean),
    ## compute confidence interval coverage (% of sims where true value
```

```
inside)
    covg_mem = mean(mean_ci1_mem <= mu_x & mu_x <= mean_ci2_mem),
    covg_naive = mean(mean_ci1_naive <= mu_x & mu_x <= mean_ci2_naive),
    covg_hybrid = mean(mean_ci1_hybrid <= mu_x & mu_x <= mean_ci2_hybrid),
    covg_signal = mean(mean_ci1_signal <= mu_x & mu_x <= mean_ci2_signal),
    ## compute confidence interval widths
    ci_width_hybrid = mean(mean_ci2_hybrid - mean_ci1_hybrid),
    ci_width_naive = mean(mean_ci2_naive- mean_ci1_naive),
    ci_width_mem = mean(mean_ci2_mem - mean_ci1_mem),
    ci_width_signal = mean(mean_ci2_signal - mean_ci1_signal)
  ) %>%
  ungroup() %>%
  ## extra variables for plotting
  mutate(sig2e_str = 'True ME Variance', sig2x_str = 'True Signal Variance',
         n_str = 'Sample Size', pct_rep = 100 - 100*pctJ)

## save summary data
saveRDS(res_long_summary_J3, 'ME_figure_data_summary_J=3.Rdata')
```

04_make_simulation_figures.R

- create ggplot2 graphics for main paper and SI
- save to image files using ggsave

```
## GOAL: make all simulation-related plots for main paper and SI 1 &2.

# load packages -----------------------------------------------------------

library(tidyverse)
library(ggh4x)   # for multiple levels of facetting
library(ggridges) # for density plots in Fig S22
library(ggtext)
library(patchwork) # for arranging panels in figure 1

# read data ---------------------------------------------------------------

## separate data frames for J=2, J=3
## need to re-save minimal pieces
res_long <- readRDS('ME_processed_output.Rdata')
res_long_summary <- readRDS('ME_figure_data_summary.Rdata')

res_long_J3 <- readRDS('ME_processed_output_J=3.Rdata')
res_long_summary_J3 <- readRDS('ME_figure_data_summary_J=3.Rdata')

# functions and settings --------------------------------------------------

## plotting parameters
```

```r
## breaks for x-axis on plots
brks <- seq(0, 90, by = 10)
major_breaks <- c(20, 40, 60, 80, 100)
minor_breaks <- c(10, 30, 50, 70, 90)

## define line width and point size for all plots
ln_size <- 1.1
pt_size <- 2 #2.5

## parameters for saving figures (inches)
fig_height <- 5
fig_width  <- 7.86
dpi <- 100

## set ggplot theme for all plots
theme_set(theme_bw() +  theme(strip.text = element_text(face='bold')))

## define facet header colors

## fills_3x3 - 3 columnx x 3 rows
fills_3x3 <- c('white', rep('gray', 3), 'gray', 'white', 'gray','gray')
fills_3x3_new <- c('white',rep('gray',3), 'white', rep('gray', 3))

## fills_5x3 - 5 columns x 3 rows
fills_5x3 <- c('white', rep('gray', 5), 'gray', 'white', 'gray', 'gray')
fills_5x3_new <- c('white', rep('gray', 5), 'white', rep('gray',3))

## fills_3x2 (reduced) 3 columns x 2 rows
fills_3x2 <- c('white', rep('gray', 3), 'gray', 'white', 'gray')
fills_3x2_new <- c('white', rep('gray',3), 'white', rep('gray',2))

## fills_joined (joining of J=2 and J=3)
fills_joined <- c('white',rep('gray',2),'white',rep('gray',2),'gray',
                  'white',rep('gray',2))
fills_joined_new <- c(rep('white',2),rep('gray',4), 'white',rep('gray',3))

## function to allow for different nested facet colors
change_facet_colors <- function(plt, colvec){
  g <- ggplot_gtable(ggplot_build(plt))
  strip_both <- which(grepl('strip-', g$layout$name))

  k <- 1
  for (i in strip_both) {
    j <- which(grepl('rect', g$grobs[[i]]$grobs[[1]]$childrenOrder))
    g$grobs[[i]]$grobs[[1]]$children[[j]]$gp$fill <- colvec[k]
    k <- k+1
  }

  return(g)
```

```
  #grid::grid.draw(g)
}


# main paper figures ------------------------------------------------------

## combine J=2 & J=3 datasets
res_long_joined <- bind_rows(res_long_summary %>%
                               mutate(J = 'J = 2') %>%
                               filter(n < 100) %>%
                               select(J, sig2_e, sig2_x, pctJ, n, pct_rep,
                                      sig2x_str, n_str, sig2_x_est,
covg_mem),
                             res_long_summary_J3 %>%
                               mutate(J = 'J = 3') %>%
                               filter(n < 100) %>%
                               select(J, sig2_e, sig2_x, pctJ, n, pct_rep,
                                      sig2x_str, n_str, sig2_x_est,
covg_mem))

## Fig 1: MEM signal variance

## left panels: true signal variance = 3
fig1_p1 <- res_long_joined %>%
  filter(J == 'J = 2', sig2_e %in% c(0.25, 6.25),
         sig2_x %in% c(3)) %>%
  mutate(n_lab = paste('n =',n),
         sig2_x_lab = paste0('Signal Variance = ',sig2_x),
         sig2_e_lab = paste0('Measurement Error Variance = ', sig2_e)) %>%
  ggplot(aes(x = pct_rep, y = sig2_x_est, color = factor(sig2_e))) +
  geom_line(size = ln_size ) +
  geom_point(size = pt_size) +
  geom_hline(aes(yintercept = sig2_x), size = 1.2, alpha = 0.5) +
  scale_x_continuous(breaks = seq(20, 100, by = 20)) +
  ggh4x::facet_nested(
    ~ sig2_x_lab + sig2_e_lab + n_lab
  ) +
  labs(
    x = '% Single Replication',
    y = 'Estimated\nSignal Variance',
    col = 'True ME\nVariance') +
  theme(panel.grid.minor = element_blank(),
        legend.position = 'none',
        axis.text.y = ggtext::element_markdown())

## right panels: true signal variance = 9
fig1_p2 <- res_long_joined %>%
  filter(J == 'J = 2', sig2_e %in% c(0.25, 6.25),
         sig2_x %in% c(9)) %>%
```

```
  mutate(n_lab = paste('n =',n),
         sig2_x_lab = paste0('Signal Variance = ',sig2_x),
         sig2_e_lab = paste0('Measurement Error Variance = ', sig2_e)) %>%
  ggplot(aes(x = pct_rep, y = sig2_x_est, color = factor(sig2_e))) +
  geom_line(size = ln_size) +
  geom_point(size = pt_size) +
  geom_hline(aes(yintercept = sig2_x), size = 1.2, alpha = 0.5) +
  scale_x_continuous(breaks = seq(20, 100, by = 20)) +
  ggh4x::facet_nested(
    ~ sig2_x_lab + sig2_e_lab + n_lab
  ) +
  labs(
    x = '% Single Replication',
    y = 'Estimated\nSignal Variance',
    col = 'True ME\nVariance') +
  theme(panel.grid.minor = element_blank(),
        legend.position = 'none',
        axis.text.y = ggtext::element_markdown())

## arrange plots side-by-side
fig1_p1 + fig1_p2 + plot_layout(guides = 'collect')

## save as png image
ggsave(plot = last_plot(),
       filename = 'Fig_01_wide.png', device='png',
       width = 1.5*fig_width, height = 0.6*fig_height,
       dpi = dpi, units='in')

## Figure 2: MEM CI Coverage, J=2 & J=3
fig_2 <- res_long_joined %>%
  mutate(n_lab = paste('n =',n)) %>%
  ggplot(aes(x = pct_rep, y = covg_mem, color = factor(sig2_e))) +
  geom_line(size = ln_size) +
  geom_point(size = pt_size) +
  geom_hline(aes(yintercept = 0.95), size = 1.2, alpha = 0.5) +
  scale_x_continuous(breaks = seq(10, 100, by = 10)) +
  scale_y_continuous(limits = c(0.915, 0.975),
                     breaks = seq(0.92,0.97,by=0.01),
                     expand = c(0,0)) +
  ggh4x::facet_nested(
    sig2x_str + sig2_x ~ J + n_lab,
    scales = 'free'
  ) +
  labs(
    x = '% Replicated',
    y = 'Empirical Coverage Proportion',
    col = 'True ME\nVariance') +
  theme(panel.grid.minor = element_blank(),
        plot.title = element_text(hjust = 0.5, face = 'bold'),
        axis.title = element_text(size = 16),
```

```
        legend.position = 'right')

fig_2 <- change_facet_colors(fig_2, fills_joined)#fills_3x3_tmp)
ggsave(filename = 'Fig_02_MEM_ci_coverage.png',
       fig_2, device = 'png', height = 1.25*fig_height,
       width = 1.5*fig_width,
       units = 'in')


# Supporting Info figures ------------------------------------------------

## Figure S9: MEM singular fit proportion, J=2
fig_S09 <- res_long_summary %>%
  ggplot(aes(x = pct_rep, y = pct_warn, color = factor(sig2_x),
             shape = factor(sig2_x))) + #, size=factor(sig2_x))) +
  geom_line(size = ln_size) +
  geom_point(size = pt_size) +
  ggh4x::facet_nested(n_str + n ~ sig2e_str + sig2_e) +
  scale_shape_manual(values = c('3' = 17, '9' = 19, '15' = 0)) +
  scale_y_continuous(limits = c(0, 0.32), breaks=seq(0,0.32,by=0.08)) +
  scale_x_continuous(breaks = major_breaks,
                     minor_breaks = minor_breaks) +
  labs(x = "% Single Replication", y = 'Proportion', color = 'Signal
Variance',
       shape = 'Signal Variance', size='Signal Variance')
fig_S09 <- change_facet_colors(fig_S09, fills_5x3_new)
plot(fig_S09)

ggsave(plot = fig_S09,
       filename = 'Fig_S09_pct_singular_fit_J=2.png',
       device = 'png', dpi = dpi, height = fig_height, width = fig_width,
units = 'in')

## Figure S10: MEM singular fit proportion, J=3
fig_S10 <- res_long_summary_J3 %>%
  ggplot(aes(x = pct_rep, y = pct_warn, color = factor(sig2_x),
             shape = factor(sig2_x))) +
  geom_line(size = ln_size) +
  geom_point(size = 1.2*pt_size) +
  ggh4x::facet_nested(n_str + n ~ sig2e_str + sig2_e) +
  scale_shape_manual(values = c('3' = 17, '9' = 19, '15' = 0)) +
  scale_y_continuous(limits = c(0, 0.32), breaks=seq(0,0.32,by=0.08)) +
  scale_x_continuous(breaks = major_breaks,
                     minor_breaks = minor_breaks) +
  labs(x = "% Double Replication", y = 'Proportion', color = 'Signal
Variance',
       shape = 'Signal Variance')
fig_S10 <- change_facet_colors(fig_S10, fills_5x3_new)
plot(fig_S10)
```

```r
ggsave(plot = fig_S10,
       filename = 'Fig_S10_pct_singular_fit_J=3.png',
       device = 'png', dpi = dpi, height = fig_height,
       width = fig_width, units = 'in')


## Figure S11: Naive mean, J=2
fig_S11 <- res_long_summary %>%
  ggplot(aes(x = pct_rep, y = mean_est_naive, color = factor(sig2_x))) +
  geom_line(size = ln_size) +
  geom_point(size = pt_size) +
  geom_hline(aes(yintercept = 10), alpha = 0.5) +
  scale_x_continuous(breaks = major_breaks,
                     minor_breaks = minor_breaks) +
  ggh4x::facet_nested(n_str + n ~ sig2e_str + sig2_e, nest_line = T) +
  labs(color = "True Signal\nVariance", y = "Estimated Mean",
       x = "% Single Replication") +
  ylim(9.9, 10.1)

fig_S11 <- change_facet_colors(fig_S11, fills_5x3_new)
ggsave(plot = fig_S11,
       filename = 'Fig_S11_Naive_mean_J=2.png',
       device = 'png', dpi = dpi, height = fig_height,
       width = fig_width, units = 'in')

## Figure S12: Hybrid mean, J=2
fig_S12 <- res_long_summary %>%
  ggplot(aes(x = pct_rep, y = mean_est_hybrid, color = factor(sig2_x))) +
  geom_line(size = ln_size) +
  geom_point(size = pt_size) +
  geom_hline(aes(yintercept = 10), alpha = 0.5) +
  scale_x_continuous(breaks = major_breaks,
                     minor_breaks = minor_breaks) +
  ggh4x::facet_nested(n_str + n ~ sig2e_str + sig2_e,
                      scales = 'free', nest_line = T) +
  labs(x = '% Single Replication', y = 'Sample Average',
       color = 'True Signal\nVariance') +
  ylim(9.9, 10.1)

fig_S12 <- change_facet_colors(fig_S12, fills_5x3_new)
ggsave(plot = fig_S12,
       filename = 'Fig_S12_Hybrid_mean_J=2.png',
       device = 'png', dpi = dpi, height = fig_height,
       width = fig_width, units = 'in')

## Figure S13: MEM mean, J=2
fig_S13 <- res_long_summary %>%
  ggplot(aes(x = pct_rep, y = mean_est, color = factor(sig2_x))) +
  geom_line(size = ln_size) +
```

```
  geom_point(size = pt_size) +
  geom_hline(aes(yintercept = 10), alpha = 0.5) +
  scale_x_continuous(breaks = major_breaks,
                     minor_breaks = minor_breaks) +
  ggh4x::facet_nested(n_str + n ~ sig2e_str + sig2_e, nest_line = T) +
  labs(color = "True Signal\nVariance", y = "Estimated Mean",
       x = "% Single Replication") +
  ylim(9.9, 10.1)

fig_S13 <- change_facet_colors(fig_S13, fills_5x3_new)
ggsave(plot = fig_S13,
       filename = 'Fig_S13_MEM_mean_J=2.png',
       device = 'png', dpi = dpi, height = fig_height,
       width = fig_width, units = 'in')

## Figure S14: Naive variance, J=2
fig_S14 <- res_long_summary %>%
  ggplot(aes(x = pct_rep, y = var_est_naive, color = factor(sig2_e))) +
  geom_line(size = ln_size) +
  geom_point(size = pt_size) +
  scale_x_continuous(breaks = seq(10, 100, by = 10), minor_breaks = NULL) +
  ggh4x::facet_nested(sig2x_str + sig2_x ~ n_str + n, scales='free') +
  labs(x='% Single Replication', y='Estimated Variance',
       color='True ME\nVariance')

fig_S14 <- change_facet_colors(fig_S14, fills_3x3_new)
ggsave(plot = fig_S14,
       filename = 'Fig_S14_Naive_variance_J=2.png',
       device = 'png', dpi = dpi, height = fig_height,
       width = fig_width, units = 'in')

## Figure S15: Hybrid variance, J=2
fig_S15 <- res_long_summary %>%
  ggplot(aes(x = pct_rep, y = var_est_hybrid, color = factor(sig2_e))) +
  geom_line(size = ln_size) +
  geom_point(size = pt_size) +
  scale_x_continuous(breaks = seq(10, 100, by = 10), minor_breaks = NULL) +
  ggh4x::facet_nested(sig2x_str + sig2_x ~ n_str + n ,
                      scales='free', nest_line = T) +
  labs(x = '% Single Replication', y = 'Estimated Variance',
       color = 'True ME\nVariance')


fig_S15 <- change_facet_colors(fig_S15, fills_3x3_new)
ggsave(plot = fig_S15,
       filename = 'Fig_S15_Hybrid_variance_J=2.png',
       device = 'png', dpi = dpi, height = fig_height,
       width = fig_width, units = 'in')
```

```r
## Figure S16: MEM signal variance, J=2
fig_S16 <- res_long_summary %>%
  ggplot(aes(x = pct_rep, y = sig2_x_est, color = factor(sig2_e))) +
  #geom_hline(aes(yintercept = sig2_x), size=1.2) +
  geom_line(size = ln_size) +
  geom_point(size = pt_size) +
  geom_hline(aes(yintercept = sig2_x), size=1.2, alpha=0.5) +
  scale_x_continuous(breaks = seq(10, 100, by = 10), minor_breaks = NULL) +
  ggh4x::facet_nested(sig2x_str + sig2_x ~ n_str + n ,
                      scales = 'free',
                      nest_line = T) +
  ggh4x::facetted_pos_scales(y = list(
    sig2_x == 3 ~ scale_y_continuous(limits = c(2.7,6),
                                     labels = c('**3**',4,5,6),
                                     breaks=c(3,4,5,6)
    ),
    sig2_x == 9 ~ scale_y_continuous(limits = c(8.5, 11),
                                     labels = c(8.5,'**9**',9.5,10,10.5,11),
                                     breaks=c(8.5, 9,9.5,10,10.5,11)),
    sig2_x == 15 ~ scale_y_continuous(limits = c(14.6, 16.6),
                                      labels = c('**15.0**',15.5,16,16.5),
                                      breaks=c(15,15.5,16,16.5))
  )) +
  labs(x = '% Single Replication', y = 'Estimated Signal Variance',
       color = 'True ME\nVariance') +
  theme(panel.grid.minor = element_blank(),
        axis.text.y = ggtext::element_markdown())


fig_S16 <- change_facet_colors(fig_S16, fills_3x3_new)
ggsave(plot = fig_S16,
       filename = 'Fig_S16_MEM_signal_variance_J=2.png',
       device = 'png', dpi = dpi, height = fig_height,
       width = fig_width, units = 'in')


## Figure S17: Naive CI coverage, J=2
fig_S17 <- res_long_summary %>%
  ggplot(aes(x = pct_rep, y = covg_naive, color = factor(sig2_x))) +
  geom_line(size = ln_size) +
  geom_point(size = pt_size) +
  geom_hline(aes(yintercept = 0.95), alpha = 0.5) +
  scale_x_continuous(breaks = major_breaks,
                     minor_breaks = minor_breaks) +
  scale_y_continuous(limits = c(0.7, 1), breaks=seq(0.7, 1, by=0.05),
                     minor_breaks = NULL) +
  ggh4x::facet_nested(n_str + n ~ sig2e_str + sig2_e) +
  labs(color = "True Signal\nVariance", y = "Empirical Coverage Proportion",
```

```
       x = "% Single Replication")

fig_S17 <- change_facet_colors(fig_S17, fills_5x3_new)
ggsave(plot = fig_S17,
       filename = 'Fig_S17_Naive_CI_coverage_J=2.png',
       device = 'png', dpi = dpi, height = fig_height,
       width = fig_width, units = 'in')

## Figure S18: Hybrid CI coverage, J=2
fig_S18 <- res_long_summary %>%
  ggplot(aes(x = pct_rep, y = covg_hybrid, color = factor(sig2_x))) +
  geom_hline(aes(yintercept = 0.95)) +
  geom_line(size = ln_size) +
  geom_point(size = pt_size) +
  scale_x_continuous(breaks = major_breaks,
                     minor_breaks = minor_breaks) +
  scale_y_continuous(limits = c(0.7, 1), breaks=seq(0.7, 1, by=0.05),
                     minor_breaks = NULL) +
  ggh4x::facet_nested(n_str + n ~ sig2e_str + sig2_e) +
  labs(color = "True Signal\nVariance", y = "Empirical Coverage Proportion",
       x = "% Single Replication")

fig_S18 <- change_facet_colors(fig_S18, fills_5x3_new)
ggsave(plot = fig_S18,
       filename = 'Fig_S18_Hybrid_CI_coverage_J=2.png',
       device = 'png', dpi = dpi, height = fig_height,
       width = fig_width, units = 'in')

## Figure S19: MEM observed CI coverage, J=2
fig_S19 <- res_long_summary %>%
  drop_na() %>%
  ggplot(aes(x = pct_rep, y = covg_mem, color = factor(sig2_x))) +
  geom_line(size = ln_size) +
  geom_point(size = pt_size) +
  geom_hline(aes(yintercept = 0.95), alpha = 0.5) +
  scale_x_continuous(breaks = major_breaks,
                     minor_breaks = minor_breaks) +
  scale_y_continuous(limits = c(0.7, 1), breaks=seq(0.7, 1, by=0.05),
                     minor_breaks = NULL) +
  ggh4x::facet_nested(n_str + n ~ sig2e_str + sig2_e) +
  labs(color = "True Signal\nVariance",
       y = "Empirical Coverage Proportion",
       x = "% Single Replication")

fig_S19 <- change_facet_colors(fig_S19, fills_5x3_new)
ggsave(plot = fig_S19,
       filename = 'Fig_S19_MEM_CI_coverage_J=2.png',
       device = 'png', dpi = dpi, height = fig_height,
       width = fig_width, units = 'in')
```

```r
## Figure S20: Diff b/w MEM and Naive CI width, J=2
fig_S20 <- res_long %>%
  drop_na() %>%
  filter(n %in% c(20, 100), sig2_e %in% c(0.25, 2.25, 6.25)) %>%
  mutate(sig2e_str = 'ME Variance', sig2x_str = 'Signal Variance',
         n_str = 'Sample Size', pct_rep = 100 - 100*pctJ) %>%
  ggplot(aes(x = (mean_ci2_naive - mean_ci1_naive) - (mean_ci2_mem -
mean_ci1_mem),
             y = interaction(pct_rep, sig2_x, sep=", "))) +
  geom_density_ridges(rel_min_height = 0.005,
                      aes(fill=factor(sig2_x))) + # , alpha=pct_rep)) +
  geom_vline(aes(xintercept=0)) +
  ggh4x::facet_nested(n_str + n ~ sig2e_str + sig2_e) +
  labs(x = 'Difference in CI Widths (Naive - MEM)',
       y = '% Replicated, Signal Var', fill='Signal\nVariance') +
  scale_y_discrete(breaks=c('100, 3','90, 3','80, 3','70, 3','60, 3','50,
3','40, 3','30, 3','20, 3','10, 3',
                            '100, 9','90, 9','80, 9','70, 9','60, 9','50,
9','40, 9','30, 9','20, 9','10, 9',
                            '100, 15','90, 15','80, 15','70, 15','60,
15','50, 15','40, 15','30, 15','20, 15','10, 15'),
                   labels=c('Balanced','','','','','50%','','','','10%',
                            'Balanced','','','','', '50%','','','', '10%',
                            'Balanced','','','','', '50%','','','', '10%')) +
  NULL


## plot name ci_diff_density_flipped_J=2
fig_S20 <- change_facet_colors(fig_S20, fills_3x2_new)
ggsave(plot = fig_S20,
       filename = 'Fig_S20_ci_diff_naive_mem.png',
       device = 'png', dpi = dpi, height = fig_height,
       width = fig_width, units = 'in')


## Figure S21: Naive mean, J=3
fig_S21 <- res_long_summary_J3 %>%
  ggplot(aes(x = pct_rep, y = mean_est_naive, color = factor(sig2_x))) +
  geom_hline(aes(yintercept = mu_x)) +
  geom_line(size = ln_size) +
  geom_point(size = pt_size) +
  scale_x_continuous(breaks = major_breaks,
                     minor_breaks = minor_breaks) +
  ggh4x::facet_nested(n_str + n ~ sig2e_str + sig2_e, nest_line = T) +
  labs(color = "True Signal\nVariance", y = "Estimated Mean",
       x = "% Double Replication") +
  ylim(9.9, 10.1)
```

```r
fig_S21 <- change_facet_colors(fig_S21, fills_5x3_new)
ggsave(plot = fig_S21,
       filename = 'Fig_S21_Naive_mean_J=3.png',
       device = 'png', dpi = dpi, height = fig_height,
       width = fig_width, units = 'in')

## Figure S22: Hybrid mean, J=3
fig_S22 <- res_long_summary_J3 %>%
  ggplot(aes(x = pct_rep, y = mean_est_hybrid, color = factor(sig2_x))) +
  geom_hline(aes(yintercept = 10)) +
  geom_line(size = ln_size) +
  geom_point(size = pt_size) +
  scale_x_continuous(breaks = major_breaks,
                     minor_breaks = minor_breaks) +
  ggh4x::facet_nested(n_str + n ~ sig2e_str + sig2_e,
                      scales = 'free', nest_line = T) +
  labs(x = '% Double Replication', y = 'Sample Average',
       color = 'ME Variance') +
  ylim(9.9, 10.1)

fig_S22 <- change_facet_colors(fig_S22, fills_5x3_new)
ggsave(plot = fig_S22,
       filename = 'Fig_S22_Hybrid_mean_J=3.png',
       device = 'png', dpi = dpi, height = fig_height,
       width = fig_width, units = 'in')

## Figure S23: MEM mean, J=3
fig_S23 <- res_long_summary_J3 %>%
  ggplot(aes(x = pct_rep, y = mean_est, color = factor(sig2_x))) +
  geom_hline(aes(yintercept = mu_x)) +
  geom_line(size = ln_size) +
  geom_point(size = pt_size) +
  scale_x_continuous(breaks = major_breaks,
                     minor_breaks = minor_breaks) +
  ggh4x::facet_nested(n_str + n ~ sig2e_str + sig2_e, nest_line = T) +
  labs(color = "True Signal\nVariance", y = "Estimated Mean",
       x = "% Double Replication") +
  ylim(9.9, 10.1)

fig_S23 <- change_facet_colors(fig_S23, fills_5x3_new)
ggsave(plot = fig_S23,
       filename = 'Fig_S23_MEM_mean_J=3.png',
       device = 'png', dpi = dpi, height = fig_height,
       width = fig_width, units = 'in')

## Figure S24: Naive variance, J=3
fig_S24 <- res_long_summary_J3 %>%
  ggplot(aes(x = pct_rep, y = var_est_naive, color = factor(sig2_e))) +
```

```
  geom_line(size = ln_size) +
  geom_point(size = pt_size) +
  scale_x_continuous(breaks = seq(10, 100, by = 10), minor_breaks = NULL) +
  ggh4x::facet_nested(sig2x_str + sig2_x ~ n_str + n, scales='free') +
  labs(x = '% Double Replication', y = 'Estimated Variance',
       color = 'True ME\nVariance')

fig_S24 <- change_facet_colors(fig_S24, fills_3x3_new)
ggsave(plot = fig_S24,
       filename = 'Fig_S24_Naive_variance_J=3.png',
       device = 'png', dpi = dpi, height = fig_height,
       width = fig_width, units = 'in')

## Figure S25: Hybrid variance, J=3
fig_S25 <- res_long_summary_J3 %>%
  ggplot(aes(x = pct_rep, y = var_est_hybrid, color = factor(sig2_e))) +
  geom_line(size = ln_size) +
  geom_point(size = pt_size) +
  scale_x_continuous(breaks = seq(10, 100, by = 10),
                     minor_breaks = NULL) +
  ggh4x::facet_nested(sig2x_str + sig2_x ~ n_str + n ,
                      scales='free', nest_line = T) +
  labs(x = '% Double Replication', y = 'Estimated Variance',
       color = 'True ME\nVariance')

fig_S25 <- change_facet_colors(fig_S25, fills_3x3_new)
ggsave(plot = fig_S25,
       filename = 'Fig_S25_Hybrid_variance_J=3.png',
       device = 'png', dpi = dpi, height = fig_height,
       width = fig_width, units = 'in')

## Figure S26: MEM signal variance, J=3
fig_S26 <- res_long_summary_J3 %>%
  ggplot(aes(x = pct_rep, y = sig2_x_est, color = factor(sig2_e))) +
  geom_line(size = ln_size) +
  geom_point(size = pt_size) +
  geom_hline(aes(yintercept = sig2_x), size = 1.2, alpha = 0.5) +
  scale_x_continuous(breaks = seq(10, 100, by = 10), minor_breaks = NULL) +
  ggh4x::facet_nested(sig2x_str + sig2_x ~ n_str + n ,
                      scales='free', nest_line = T) +
  ggh4x::facetted_pos_scales(y = list(
    sig2_x == 3 ~ scale_y_continuous(limits = c(2.7,6), breaks=c(3,4,5,6)),
    sig2_x == 9 ~ scale_y_continuous(limits = c(8.5, 11),
                                     breaks=c(8.5, 9,9.5,10,10.5,11)),
    sig2_x == 15 ~ scale_y_continuous(limits = c(14.6, 16.6),
breaks=c(15,15.5,16,16.5))
  )) +
  labs(x = '% Double Replication', y = 'Estimated Signal Variance',
       color = 'True ME\nVariance') +
```

```
  theme(panel.grid.minor = element_blank())


fig_S26 <- change_facet_colors(fig_S26, fills_3x3_new)
ggsave(plot = fig_S26,
       filename = 'Fig_S26_MEM_signal_variance_J=3.png',
       device = 'png', dpi = dpi, height = fig_height,
       width = fig_width, units = 'in')

## Figure S27: Naive CI coverage, J=3
fig_S27 <- res_long_summary_J3 %>%
  ggplot(aes(x = pct_rep, y = covg_naive, color = factor(sig2_x))) +
  geom_line(size = ln_size) +
  geom_point(size = pt_size) +
  geom_hline(aes(yintercept = 0.95), alpha = 0.5) +
  scale_x_continuous(breaks = major_breaks,
                     minor_breaks = minor_breaks) +
  scale_y_continuous(limits = c(0.7, 1), breaks = seq(0.7,1,by=0.05),
                     minor_breaks = NULL) +
  ggh4x::facet_nested(n_str + n ~ sig2e_str + sig2_e) +
  labs(color = "True Signal\nVariance", y = "Empirical Coverage Proportion",
       x = "% Double Replication")

fig_S27 <- change_facet_colors(fig_S27, fills_5x3_new)
ggsave(plot = fig_S27,
       filename = 'Fig_S27_Naive_CI_coverage_J=3.png',
       device = 'png', dpi = dpi, height = fig_height,
       width = fig_width, units = 'in')

## Figure S28: Hybrid CI coverage, J=3
fig_S28 <- res_long_summary_J3 %>%
  ggplot(aes(x = pct_rep, y = covg_hybrid, color = factor(sig2_x))) +
  geom_hline(aes(yintercept = 0.95)) +
  geom_line(size = ln_size) +
  geom_point(size = pt_size) +
  scale_x_continuous(breaks = major_breaks,
                     minor_breaks = minor_breaks) +
  scale_y_continuous(limits = c(0.7, 1), breaks = seq(0.7,1,by=0.05),
                     minor_breaks = NULL) +
  ggh4x::facet_nested(n_str + n ~ sig2e_str + sig2_e) +
  labs(color = "True Signal\nVariance", y = "Empirical Coverage Proportion",
       x = "% Double Replication")

fig_S28 <- change_facet_colors(fig_S28, fills_5x3_new)
ggsave(plot = fig_S28,
       filename = 'Fig_S28_Hybrid_CI_coverage_J=3.png',
       device = 'png', dpi = dpi, height = fig_height,
       width = fig_width, units = 'in')
```

```r
## Figure S29: MEM observed CI coverage, J=3
fig_S29 <- res_long_summary_J3 %>%
  drop_na() %>%
  ggplot(aes(x = pct_rep, y = covg_mem, color = factor(sig2_x))) +
  geom_line(size = ln_size) +
  geom_point(size = pt_size) +
  geom_hline(aes(yintercept = 0.95), alpha = 0.5) +
  scale_x_continuous(breaks = major_breaks,
                     minor_breaks = minor_breaks) +
  scale_y_continuous(limits = c(0.7, 1), breaks = seq(0.7,1,by=0.05),
                     minor_breaks = NULL) +
  ggh4x::facet_nested(n_str + n ~ sig2e_str + sig2_e) +
  labs(color = "True Signal\nVariance",
       y = "Empirical Coverage Proportion",
       x = "% Double Replication")

fig_S29 <- change_facet_colors(fig_S29, fills_5x3_new)
ggsave(plot = fig_S29,
       filename = 'Fig_S29_MEM_CI_coverage_J=3.png',
       device = 'png', dpi = dpi, height = fig_height,
       width = fig_width, units = 'in')

## Figure S30: MEM signal CI coverage, J=2
fig_S30 <- res_long_summary %>%
  drop_na() %>%
  ggplot(aes(x = pct_rep, y = covg_signal, color = factor(sig2_x))) +
  geom_line(size = ln_size) +
  geom_point(size = pt_size) +
  geom_hline(aes(yintercept = 0.95), alpha = 0.5) +
  scale_x_continuous(breaks = major_breaks,
                     minor_breaks = minor_breaks) +
  scale_y_continuous(limits = c(0.7, 1), breaks = seq(0.7, 1, by=0.05),
                     minor_breaks = NULL) +
  ggh4x::facet_nested(n_str + n ~ sig2e_str + sig2_e) +
  labs(color = "True Signal\nVariance",
       y = "Empirical Coverage Proportion",
       x = "% Double Replication"
  ) +
  theme()

fig_S30 <- change_facet_colors(fig_S30, fills_5x3_new)
ggsave(plot = fig_S30,
       filename = 'Fig_S30_Signal_CI_coverage_J=2.png',
       device = 'png', dpi = dpi, height = fig_height,
       width = fig_width, units = 'in')
```

Case Study Code - R

Case Study 1: NHEXAS Arsenic data

```r
# load libraries and data ------------------------------------------------

library(readxl)    # for reading in data, v1.3.1
library(tidyverse) # for reading and manipulating data, v1.3.0
library(lme4)      # for fitting MEM model, v1.1-25
library(lmerTest)  # for Kenward-Roger CIs, v3.1.3
library(EnvStats)  # for goodness-of-fit testing, v2.4.0

# data manipulation ------------------------------------------------------

## read in CSV
df_arsenic <- read_xlsx('es3c02231_si_003.xlsx',
                        sheet = 'CS1 NHEXAS Arsenic Data') %>%
  rename(PARID = 'Participant ID', REPFLG = 'Replicate Flag',
         conc_orig = 4, conc_repl = 6)

head(df_arsenic)
## PARID is our grouping variable
## (bc Sample IDs are different b/w orig and replicate)
## REPFLG is a binary indicating whether a sample has a replicate

## convert to long format for lme4 model fitting
df_arsenic_long <- df_arsenic %>%
  # keep ID column and two conc msmts
  select(PARID, conc_orig, conc_repl) %>%
  ## switch from wide to long format
  pivot_longer(conc_orig:conc_repl, values_to='conc', names_to=NULL) %>%
  # -999 is their code for missing data
  filter(conc != -999)

## add rank variable and sort
df_arsenic_ranked <- df_arsenic %>%
  ## sort and add rank column
  mutate(conc_repl = na_if(conc_repl, -999)) %>%
  ## sort by original conc, replicate conc
  arrange(conc_orig, conc_repl) %>%
  add_column(field_rank = 1:179)

# model fitting ----------------------------------------------------------

## fit random effects model
fit.mem <- lmer(conc ~ (1 | PARID), data = df_arsenic_long, REML = TRUE)
```

```
## print model summary
summary(fit.mem)

## extract model coefficients (mean, signal var, ME var)
est_mem <- c(fit.mem@beta, as.data.frame(VarCorr(fit.mem))$vcov)

## compute 95% confidence interval for observation mean
mem_kr <- contest(fit.mem, L = 1, joint = FALSE, confint = TRUE,
                  level = 0.95, ddf = 'Kenward-Roger')

ci_mem <- c(mem_kr$lower, mem_kr$upper)

# run naive and hybrid ---------------------------------------------------

## Naive approach - original data only
Ysamp_naive <- df_arsenic$conc_orig

nom <- 0.05
nnaive <- length(Ysamp_naive)

## compute Naive mean, variance, CI
est_naive <- c(mean(Ysamp_naive), var(Ysamp_naive), NA)
ci_naive <- mean(Ysamp_naive) + c(-1,1) *
  qt(1-nom/2, df=nnaive-1) *
  sqrt(var(Ysamp_naive)/nnaive)

## Hybrid approach - all data
Ysamp_hybrid <- df_arsenic_long$conc

nhybrid <- length(Ysamp_hybrid)

## compute Hybrid mean, variance, CI
est_hybrid <- c(mean(Ysamp_hybrid), var(Ysamp_hybrid), NA)
ci_hybrid <- mean(Ysamp_hybrid) + c(-1,1) *
  qt(1-nom/2, df=nhybrid-1) *
  sqrt(var(Ysamp_hybrid)/nhybrid)

# plots ------------------------------------------------------------------

## Figure S2: goodness of fit for orig msmts
gof <- gofTest(df_arsenic$conc_orig)
plot(gof)

## Figure S3: plot of concentrations, ordered by rank
ggplot(df_arsenic_ranked) +
  geom_segment(aes(x=field_rank, xend=field_rank,
                   y=conc_orig, yend=conc_repl), col='gray') +
  geom_point(aes(field_rank, conc_orig), col='blue') +
  geom_point(aes(field_rank, conc_repl), col='orange') +
```

```
  scale_x_continuous(limits = c(1, 179), breaks = seq(0,180,by=10),
                     expand = c(0.01, 0.01)) +
  theme_bw() +
  labs(x='Observation ranked', y=expression(paste("As Concentration (", mu,
"g/L)")),
       title='Arsenic Measurements',
       subtitle = 'Original measurements in blue, Replicates in orange'
       #subtitle = 'ordered from lowest to highest original msmt'
       )

## scatterplot of original vs replicates
df_arsenic %>%
  filter(REPFLG == 1) %>%
  ggplot(aes(conc_orig, conc_repl)) +
  geom_abline(aes(slope=1, intercept=0)) +
  geom_point(col='#00A65A') +
  labs(x='Original Concentration', y='Replicate Concentration') +
  coord_equal() +
  theme_bw()


# bootstrap for signal mean ------------------------------------------------

## define function
run_bootstrap <- function(fit_ss, fit_mean,
                          fit_signal_var, B=10000, seed=1234){

  ## matrix for storing output
  boot_stats <- matrix(NA, nrow=B, ncol=2)
  colnames(boot_stats) <- c('Xbar','varx')

  set.seed(seed)
  ## repeat bootstrap B times, return sample mean and variance
  for(b in 1:B){
    samp_cur <- rnorm(n = fit_ss, mean = fit_mean, sd = sqrt(fit_signal_var))

    Xbar <- mean(samp_cur)
    varx <- var(samp_cur)

    boot_stats[b,] <- c(Xbar, varx)
  }

  return(as.data.frame(boot_stats))
}


## run bootstrap for Signal CI (10,000 draws)
boot_ars <- run_bootstrap(fit_ss = 179, fit_mean = est_mem[1],
                          fit_signal_var = est_mem[2], B=10000)
```

```r
## compute 95% percentile CI
quantile(boot_ars$Xbar, c(0.025, 0.975))

## Figure S4: histogram of boostrap distribution
hist(boot_ars$Xbar, main = 'Histogram of 10,000 As Bootstrap Sample Signal
Means',
     xlab = expression(paste("As Signal Means (", mu, "g/L)")),
     xlim = range(boot_ars$Xbar),
     ylab = 'Frequency')

abline(v = quantile(boot_ars$Xbar, c(0.025, 0.975)), col = 'red')

# summary stats for Table 1 ------------------------------------------------
run_t_int <- function(mn, vr, n, exp=FALSE, level = 0.95){

  res <- mn + c(-1,1) * qt(level + (1-level)/2, df = n-1) * sqrt(vr/n)

  if(exp){
    res <- exp(res)
  }

  res
}

## naive estimates
est_naive[1] # mean
est_naive[2] # variance

ci_naive       # CI
diff(ci_naive) # CI width

100 * sqrt(est_naive[2]) / est_naive[1] # RSD


## hybrid estimates
est_hybrid[1] # mean
est_hybrid[2] # variance

ci_hybrid       # CI
diff(ci_hybrid) # CI width

100 * sqrt(est_hybrid[2]) / est_hybrid[1] # RSD

## MEM estimates
est_mem[1] # mean
est_mem[2] # signal variance
est_mem[3] # measurement error variance
```

```
ci_mem       # CI
diff(ci_mem) # CI width

# RSD_x: 100 * sqrt(signal_var) / mean_est
100 * sqrt(est_mem[2]) / est_mem[1]

# ICC: 100 * signal_var / (signal_var + ME_var)
100 * est_mem[2] / (est_mem[2] + est_mem[3])

## bootstrap interval for signal mean
bint <- quantile(boot_ars$Xbar, c(0.025, 0.975))
bint
diff(bint)

## t-interval for signal mean
tint <- run_t_int(mn = est_mem[1], vr = est_mem[2], n = 179)
tint
diff(tint)
```

Case Study 2: NHEXAS Chromium data

```
# load packages ------------------------------------------------------------

library(tidyverse)
library(readxl)
library(lme4)
library(lmerTest)
library(EnvStats)
library(ggpubr)

# read data ----------------------------------------------------------------

df_chromium <- read_xlsx('es3c02231_si_003.xlsx',
                  sheet = 'CS2 NHEXAS Chromium Data', range='A1:I29') %>%
  select(Obs:SAMPTYPE) %>%
  mutate(PARID = factor(PARID))

names(df_chromium)

## add rank variable and sort
df_chromium_ranked <- df_chromium %>%
  ## sort by original conc, replicate conc
  arrange(lnCr_S, lnCr_R) %>%
  add_column(field_rank = 1:28)
```

```r
## paired t-test on natural log scale - no sig diff
t.test(df_chromium$lnCr_S, df_chromium$lnCr_R, paired=TRUE)
# t = 1.7638, df = 27, p-value = 0.08907


## convert to long format for model fitting
df_chromium_long <- df_chromium %>%
  pivot_longer(cols = c(lnCr_S, lnCr_R), names_to = 'samp_type',
               values_to = 'lnCr')


# model fitting ------------------------------------------------------------

## fit random effects model
fit.mem.cr <- lmer(lnCr ~ (1 | PARID), REML=TRUE,
                   data=df_chromium_long)

## print model summary
summary(fit.mem.cr)

## plot residuals
plot(fitted.values(fit.mem.cr), resid(fit.mem.cr),
     main = 'Residuals vs. Fitted Values, logCr model')
abline(h=0)

## extract model coefficients (mean, signal var, ME var)
est_mem_log <- c(fit.mem.cr@beta, as.data.frame(VarCorr(fit.mem.cr))$vcov)

## compute 95% confidence interval for observation mean
mem_log_kr <- contest(fit.mem.cr, L = 1, joint = FALSE, confint = TRUE,
                  level = 0.95, ddf = 'Kenward-Roger')

ci_mem_log <- c(mem_log_kr$lower, mem_log_kr$upper)


# run naive and hybrid -----------------------------------------------------

## Naive approach - original data only
Ysamp_naive_log <- df_chromium$lnCr_S

nom <- 0.05
nnaive <- length(Ysamp_naive_log)

## compute Naive mean, variance, CI
est_naive_log <- c(mean(Ysamp_naive_log), var(Ysamp_naive_log), NA)
ci_naive_log <- mean(Ysamp_naive_log) + c(-1,1) *
  qt(1-nom/2, df=nnaive-1) *
```

```
  sqrt(var(Ysamp_naive_log)/nnaive)

## Hybrid approach - all data
Ysamp_hybrid_log <- df_chromium_long$lnCr

nhybrid <- length(Ysamp_hybrid_log)

## compute Hybrid mean, variance, CI
est_hybrid_log <- c(mean(Ysamp_hybrid_log), var(Ysamp_hybrid_log), NA)
ci_hybrid_log <- mean(Ysamp_hybrid_log) + c(-1,1) *
  qt(1-nom/2, df=nhybrid-1) *
  sqrt(var(Ysamp_hybrid_log)/nhybrid)


# plots ------------------------------------------------------------------

## Figure S2: goodness of fit for orig msmts
gof <- gofTest(df_chromium$lnCr_S)
plot(gof)

## plot of concentrations, ordered by rank
ggplot(df_chromium_ranked) +
  geom_segment(aes(x=field_rank, xend=field_rank,
                   y=lnCr_S, yend=lnCr_R), col='gray') +
  geom_point(aes(field_rank, lnCr_S), col='blue') +
  geom_point(aes(field_rank, lnCr_R), col='orange') +
  scale_x_continuous(limits = c(1, 28), breaks = 1:28,
                     labels = df_chromium_ranked$PARID,
                     expand = c(0.01, 0.01)) +
  theme_bw() +
  labs(x='Ordered Samples (PARID)', y='lnCr',
       title='Log-scale Chromium Measurements',
       subtitle = 'Original Measurements in blue, Replicates in Orange') +
  theme(axis.text.x = element_text( angle=90, vjust=0.5),
        axis.title.y = ggtext::element_markdown())

## scatterplot of original vs replicates
df_chromium %>%
  ggplot(aes(x = lnCr_S, y = lnCr_R)) +
  ## add points
  geom_point(col='orange', size = 2.5) +
  ## add y=x reference line
  geom_abline(aes(slope = 1, intercept = 0)) +
  labs(x=' log of Original Concentration',
       y='log of Replicate Concentration') +
  ggpubr::stat_cor(label.sep = '',
                   label.x = -3.5, label.y = 3.5,
                   p.digits = NA, color='orange') +
  ## fix axis limits
```

```
  coord_fixed(xlim = c(-3.75,3.75), ylim=c(-3.75,3.75)) +
  theme_bw()

# bootstrap for signal CI mean ----------------------------------------


## function to generate bootstrap sample means from parameters
run_bootstrap <- function(fit_ss, fit_mean,
                          fit_signal_var, B=10000, seed=1234){

  ## matrix for storing output
  boot_stats <- matrix(NA, nrow=B, ncol=2)
  colnames(boot_stats) <- c('Xbar','varx')

  set.seed(seed)
  ## repeat bootstrap B times, return sample mean and variance
  for(b in 1:B){
    samp_cur <- rnorm(n = fit_ss, mean = fit_mean, sd = sqrt(fit_signal_var))

    Xbar <- mean(samp_cur)
    varx <- var(samp_cur)

    boot_stats[b,] <- c(Xbar, varx)
  }

  return(as.data.frame(boot_stats))
}



## run bootstrap procedure (10,000 draws)
boot_cr_log_r <- run_bootstrap(fit_ss = 28, fit_mean = est_mem_log[1],
                               fit_signal_var = est_mem_log[2], B=10000)
## bootstrap interval on log-scale
quantile(boot_cr_log_r$Xbar, c(0.025, 0.975))

## bootstrap interval on original scale
exp(quantile(boot_cr_log_r$Xbar, c(0.025, 0.975)))


## Figure S8: histogram of bootstrap sample means
hist(exp(boot_cr_log_r$Xbar),
     main = 'Histogram of 10,000\nBootstrap Sample Cr Signal Geometric
Means',
     xlab = expression(paste("Cr Signal Geometric Means (", mu, "g/L)")) )
abline(v = exp(quantile(boot_cr_log_r$Xbar, c(0.025, 0.975))), col=2)
```

```r
# summary stats for table 1 ------------------------------------------------

run_t_int <- function(mn, vr, n, exp=FALSE, level = 0.95){
  res <- mn + c(-1,1) * qt(level + (1-level)/2, df = n-1) * sqrt(vr/n)
  if(exp){
    res <- exp(res)
  }

  res
}

## naive estimates
exp(est_naive_log[1]) # geometric mean
est_naive_log[2]      # variance (of logPb)

exp(ci_naive_log)       # CI
diff(exp(ci_naive_log)) # CI width

# %RSD: 100 * sqrt(exp(sigma^2) - 1)
100 * sqrt(exp(est_naive_log[2]) - 1)

## hybrid estimates
exp(est_hybrid_log[1]) # geometric mean
est_hybrid_log[2]      # variance (of logPb)

exp(ci_hybrid_log)       # CI
diff(exp(ci_hybrid_log)) # CI width

# %RSD: 100 * sqrt(exp(sigma^2) - 1)
100 * sqrt(exp(est_hybrid_log[2]) - 1)

## MEM estimates
exp(est_mem_log[1]) # geometric mean
est_mem_log[2]      # signal variance (of logPb)
est_mem_log[3]      # measurement error variance (of logPb)

exp(ci_mem_log)       # CI
diff(exp(ci_mem_log)) # CI width

## t-interval for signal geo. mean
tint <- run_t_int(mn = est_mem_log[1], vr = est_mem_log[2], n = 28, exp =
TRUE)
tint
diff(tint)

## bootstrap interval for signal geo. mean
bint <- quantile(boot_cr_log_r$Xbar, c(0.025, 0.975))
exp(bint)
diff(exp(bint))
```

```r
# %RSD_x: 100 * sqrt(exp(signal_var) - 1)
100 * sqrt(exp(est_mem_log[2]) - 1)

# ICC: 100 * signal_var / (signal_var + ME_var)
100 * est_mem_log[2] / (est_mem_log[2] + est_mem_log[3])
```

Case Study Code - SAS

Case Study 1: NHEXAS Arsenic data

```
%let job=Case Study 1.As;


**********;

proc import datafile="es3c02231_si_003.xlsx"
        out=Arsenic
        dbms=xlsx
        replace;
        range="CS1 NHEXAS Arsenic Data$A1:F180" ;
*        getnames=no;
run;

proc contents data=arsenic;
run;

proc print data=arsenic;
run;


**********;
data data_sample(drop=Replicate_Sample_ID Conc_Rep)
     data_replicate(drop=Original_Sample_ID Conc_Samp) ;
set arsenic(rename=(var4=Conc_Samp var6=Conc_Rep));


if Replicate_Sample_ID ne 'NA' then do; PairFlag=2; output data_replicate;
end;
PairFlag=1; output data_sample;


proc print data=data_sample;
title 'data_sample';
run;
```

```
proc print data=data_replicate;
title 'data_replicate';
run;


***********;
data As;
set data_sample(rename=(Original_Sample_ID=OrigID  Conc_Samp=Conc))
    data_replicate(rename=(Replicate_Sample_ID=OrigID  Conc_Rep=Conc)) ;

proc sort data=As;
by participant_ID PairFlag;

proc print data=As;
title 'As';
run;


************ As model random effects KENWARDROGER ddfm **********;
ods rtf file="mixed (&job) As random ddfm= KENWARDROGER.rtf";
ods graphics on;

proc mixed data=As plots=all covtest cl method=reml;
class participant_ID ;
model Conc = / solution outpm=predmeans ddfm=KENWARDROGER;
random participant_ID ;
run;

ods graphics off;
ods rtf close;


ods csv file="print (&job) As random ddfm=KENWARDROGER data=predmeans.csv";

proc print data=predmeans;
run;

ods csv close;
```

Case Study 2: NHEXAS Chromium data

```
%let job=Case Study 2.Pb;


************;
proc import file="es3c02231_si_003.xlsx"
        out=Cr
```

```
        dbms=xlsx
        replace;
        range='A1:I29';
        sheet='CS2 NHEXAS Chromium Data';
run;

ods csv file="contents data=cr.csv";

proc contents data=cr;
run;

proc print data=cr noobs;
run;

ods csv close;




************;
data Cr_orig(keep=parid lnCr_S rename=(lnCr_S = lnCr))
     Cr_rep(keep=parid lnCr_R rename=(lnCr_R = lnCr));
set cr;

run;
************;
data lnCr;
set Cr_orig Cr_rep;

proc sort data=lnCr;
by parid;


ods csv file="print (&job) data=lnCr.csv";

proc print data=lnCr;
run;

ods csv close;



************ random effects KENWARDROGER ddfm model  **********;
ods rtf file="mixed (&job) lnCr random ddfm= KENWARDROGER.rtf";
ods graphics on;

proc mixed data=lnCr plots=all covtest cl method=reml;
class parid  ;
```

```
model lnCr = / solution outpm=predmeans ddfm=KENWARDROGER ;
random parid ;
run;

ods graphics off;
ods rtf close;


ods csv file="print (&job) Cr random ddfm=KENWARDROGER data=predmeans.csv";

proc print data=predmeans;
run;

ods csv close;
```

Study Design Code

- generate lognormal datasets of various sample sizes and # of replicates
- fit Naive and MEM models to 1000 datasets per scenario
- compare CI widths of each and plot density of differences in widths

study_design_example.R

```
## load packages
library(ggridges)
library(tidyverse)
library(lme4)
library(lmerTest)

## create function for simulating study planning scenarios
run_study_plan <- function(n, nrep, nsims){

  # population log-scale mean, log(geometric mean)
  mu <- log(1)
  # signal variance log-scale
  sig2_x <- log(3)
  # measurement error variance log-scale
  sig2_eps <- log(2)

  ## fix random seed
  set.seed(123)

  ## sample labels (repeat ones with replicates)
  samp_id <- c(1:n, 1:nrep)
```

```
  ## create empty data frame for storing results
  summary_table_all <- as.data.frame(matrix(NA, ncol = 14, nrow = 2 * nsims))
  colnames(summary_table_all) <- c("sim_num","method","n_samples", "n_reps",
'n_reps_theory',

"singular","mean","variance","signal_variance", "me_variance",
                                   "confidence_interval","ci_width",
"approx_signal_ci","signal_ci_width")

  ## loop over each simulation
  for(i in 1:nsims){
    ## random effects (1 per site)
    ln_tau <- rnorm(n = n, mean = 0, sd = sqrt(sig2_x))

    ## repeat tau for replicated observations
    ln_tau <- ln_tau[samp_id]

    ## measurement errors (1 per observation)
    ln_eps <- rnorm(n = n + nrep, mean = 0, sd = sqrt(sig2_eps))

    ## observed measurements
    Y <- mu + ln_tau + ln_eps             # note Y is log-scale

    ## combine id's with observation data
    data_out <- data.frame(id = samp_id, Y = Y)

    simdata <- data.frame(id = samp_id, Y=Y, ln_tau=ln_tau, ln_eps=ln_eps)

    # estimation methods ------------------------------------------

    ## naive method
    n_naive_samples <- n
    n_naive_reps <- 0
    naive_data <- Y[1:n]
    naive_mn <- mean(naive_data)
    naive_mn_gm <- exp(naive_mn)
    naive_var <- var(naive_data)

    naive_ci <- c(naive_mn - qt(p = 0.975, df = n - 1) * sqrt(naive_var / n),
                  naive_mn + qt(p = 0.975, df = n - 1) * sqrt(naive_var / n))

    naive_ci_gm <- c(exp(naive_ci[1]),exp(naive_ci[2]))

    ## MEM method
    fit.mem <- suppressMessages(lmer(Y ~ (1 | id), data = data_out, REML =
TRUE))

    ## check if model converged (0) or had a singular fit (1)
    singular <- ifelse(is.null(fit.mem@optinfo$conv$lme4$messages),
```

```
                      0, 1)

    n_MEM_samples <- n
    n_MEM_reps <- nrep
    mem_mn <- fit.mem@beta
    mem_mn_gm <- exp(mem_mn)
    mem_vars <- as.data.frame(VarCorr(fit.mem))
    mem_signal_var <- mem_vars$vcov[1]
    mem_me_var <- mem_vars$vcov[2]

    ## MEM CI for observed mean
    mem_kr <- contest(fit.mem, ddf='Kenward-Roger', L=1,
                      confint=TRUE, rhs=0, level=0.95, joint=FALSE)
    mem_ci_kr <- c(mem_kr$lower, mem_kr$upper)

    ## exponentiated back to original scale
    mem_ci_gm_kr <- c(exp(mem_ci_kr[1]), exp(mem_ci_kr[2]))

    ## MEM CI for signal mean
    approx_signal_ci <- c(mem_mn - qt(p = 0.975, df = n - 1) *
sqrt(mem_signal_var / n),
                          mem_mn + qt(p = 0.975, df = n - 1) *
sqrt(mem_signal_var / n))

    ## exponentiated back to original scale
    approx_signal_ci_gm <-
c(exp(approx_signal_ci[1]),exp(approx_signal_ci[2]))


    ## store all estimates in summary table
    summary_table_cur <- data.frame(sim_number = i,
                                    method = c("Naive", "MEM"),
                                    n_samples = c(n_naive_samples,
n_MEM_samples),
                                    n_reps = c(n_naive_reps, n_MEM_reps),
                                    n_reps_theory = nrep,
                                    singular = c(NA, singular),
                                    mean = c(naive_mn_gm, mem_mn_gm),
                                    variance = c(naive_var,
sum(mem_signal_var, mem_me_var)),
                                    signal_variance = c(NA,  mem_signal_var),
                                    me_variance = c(NA,  mem_me_var),
                                    confidence_interval = c(
                                      paste0('[', round(naive_ci_gm[1], 3),
', ',round(naive_ci_gm[2], 3),']'),
                                      #paste0('[', round(mem_ci_gm[1], 3), ',
',round(mem_ci_gm[2], 3),']')
                                      paste0('[', round(mem_ci_gm_kr[1], 3),
', ',round(mem_ci_gm_kr[2], 3),']')
```

```
                                    ),
                                    ci_width = c(
                                      diff(naive_ci_gm),
diff(mem_ci_gm_kr[1:2])#diff(mem_ci_gm[1:2])
                                    ),
                                    approx_signal_ci = c(NA,
                                                         paste0('[',
round(approx_signal_ci_gm[1], 3), ', ',round(approx_signal_ci_gm[2], 3),']')
                                    ),
                                    signal_ci_width = c(
                                      NA, diff(approx_signal_ci_gm[1:2])
                                    )
    )
    ## add current results to overall table
    summary_table_all[(2*i - 1):(2*i), ] <- summary_table_cur
  }

  return(summary_table_all)

}

## set up study design parameters
## n = # of original samples, nrep = # of replicates
all_study_settings <- data.frame(
  n = c(rep(30,4), rep(50,4), rep(100,4), rep(300,4)),
  nreps = c(3,7,15,30, 5,12,25,50, 10,25,50,100, 30,75,150,300)
)

## create list for storing each setting's results
study_plan_results <- vector('list', length = nrow(all_study_settings))

tic <- Sys.time()

## loop over each set of parameters, with 1000 sims each
for(j in 1:nrow(all_study_settings)){
  n_cur <- all_study_settings$n[j]
  n_rep_cur <- all_study_settings$nreps[j]
  study_plan_results[[j]] <- run_study_plan(n = n_cur, nrep = n_rep_cur,
nsims=1000)

  toc <- Sys.time()
  print(paste0('Done with setting ', j, ' of ', nrow(all_study_settings),
               ': ', round(difftime(toc, tic),2), ' ', units(difftime(tic,
toc))))
}

## combine all results into one large data frame
study_plan_results_all <- do.call('rbind', study_plan_results)
```

```r
## compute CI differences, after dropping singular fits
study_plan_ci_diff <- study_plan_results_all %>%
  select(n_samples, n_reps_theory, sim_num, method, ci_width, singular) %>%
  pivot_wider(id_cols = c(n_samples, n_reps_theory, sim_num),
              names_from = method, values_from = c(ci_width, singular)) %>%
  filter(singular_MEM == 0) %>%
  mutate(ci_diff =  ci_width_MEM - ci_width_Naive)

## define names for panels in graphic
sample_names_vec <- c(
  '30' = 'n = 30',
  '50' = 'n = 50',
  '100' = 'n = 100',
  '300' = 'n = 300'
)

## plot empirical density curves
study_plan_ci_diff %>%
  ggplot() +
  ## add density curves
  geom_density_ridges_gradient(aes(x = ci_diff, y = factor(n_reps_theory),
                                   group = factor(n_reps_theory), fill =
stat(x < 0))) +
  ## add vertical line at 0
  geom_vline(aes(xintercept = 0)) +
  ## add text for MEM % smaller
  geom_text(data = df_mem_smaller, aes(x = -0.25, y = factor(n_reps_theory),
                                       label = pct_MEM_smaller),
            color = '#00bfc4', nudge_y = 0.5, fontface='bold') +
  ## add text for Naive % smaller
  geom_text(data = df_mem_smaller, aes(x = 0.25, y = factor(n_reps_theory),
                                       label = pct_Naive_smaller),
            color = '#f8766d', nudge_y = 0.5, fontface='bold') +
  ## split into panels
  facet_wrap(~n_samples, scales = 'free_y',ncol=1,
             labeller = labeller(n_samples = sample_names_vec)) +
  labs(
    x = 'Difference in CI Widths (MEM - Naive)',
    y = '# of replicates'
  ) +
  scale_y_discrete(expand = expansion(mult = c(0, 0.25))) +
  scale_fill_manual(values = c('TRUE'='#00bfc4','FALSE' = '#f8766d'),
                    labels = c('TRUE' = 'MEM','FALSE' = 'Naive')) +
  labs(fill = 'Smaller Interval') +
  theme(plot.title = element_text(hjust=0.5),
        strip.text = element_text(size=14),
        legend.position = 'left')

## save plot to png file
ggsave(plot = last_plot(), filename = 'Fig_S30_study_design_density.png',
```

```
       device = 'png', width = fig_width, height = 1.8*fig_height,
       dpi = dpi, units='in')
```

Other Files

Code for generating Figures S1, S7 - R

- graph Normal density curves corresponding to Equation (1)
- graph Lognormal density curves corresponding to Equation (6)

```
library(tidyverse)

# normal additive errors ------------------------------------------------

## set mean and variance parameters
sig2x <- 9
sig2e <- 2.25
mu <- 10

## create dataframe of normal probability density values
df <- data.frame(x = seq(-5, 25, length.out = 1000))
df <- df %>%
  mutate(signal = dnorm(x, mean = mu, sd = sqrt(sig2x)),
         me = dnorm(x, mean = 0, sd = sqrt(sig2e)),
         msmt = dnorm(x, mean = mu, sd = sqrt(sig2x + sig2e)))

## convert to long format
df_long <- df %>%
  pivot_longer(cols = c(signal, me, msmt),
               names_to = 'var', values_to = 'density') %>%
  mutate(var = factor(var, levels = c('me','signal','msmt')))


## create ggplot
p_density2 <- ggplot(df_long, aes(x = x, y = density,
                                  color = var, linetype = var)) +
  geom_vline(aes(xintercept = 0)) +
  geom_line(aes(size = var)) +
  theme_bw() +
  scale_linetype_manual(
    name = '',
    labels = c(
      'me' = 'Measurement Errors\nN(0,2.25)',
      'signal' = 'Envr. Signal\nN(10,9)',
      'msmt' = 'Observed Measurements\nN(10, 11.25)'
    ),
    values = c('me' = 'dashed', 'signal' = 'dashed', 'msmt' = 'solid')
  ) +
```

```
  scale_size_manual(
    name = '',
    labels = c(
      'me' = 'Measurement Errors\nN(0,2.25)',
      'signal' = 'Envr. Signal\nN(10,9)',
      'msmt' = 'Observed Measurements\nN(10, 11.25)'
    ),
    values = c('me' = 1, 'signal' = 1, 'msmt' = 1.2)
  ) +
  scale_color_manual(
    name = '',
    labels = c(
      'me' = 'Measurement Errors\nN(0,2.25)',
      'signal' = 'Envr. Signal\nN(10,9)',
      'msmt' = 'Observed Measurements\nN(10, 11.25)'
    ),
    values = c(
      'me' = 'green',
      'signal' = 'blue',
      'msmt' = 'red'
    )
  ) +
  labs(x = 'Concentration', y = '') +
  scale_x_continuous(limits = c(-5, 25), expand = c(0,0)) +
  scale_y_continuous(limits = c(0, 0.27), expand = c(0,0)) +
  theme(legend.position = 'top',
        legend.text.align = 0.5,
        axis.text.y = element_blank(),
        axis.ticks.y = element_blank(),
        legend.key.width = unit(1,'cm'),
        panel.grid.minor = element_blank())

p_density2

ggsave(filename = 'Fig_01_simplified.png',
       plot = p_density2, width = 6.5, height = 4, units = 'in', dpi = 300)

# lognormal multiplicative errors ----------------------------------------

## set mean and variance parameters
sig2x_ln <- 1
sig2e_ln <- 0.75
mu_ln <- 1


## create dataframe of lognormal probability density values
df_ln <- data.frame(x = seq(0, 25, length.out = 1000)) %>%
  mutate(signal = dlnorm(x, meanlog = mu_ln, sdlog = sqrt(sig2x_ln)),
         me = dlnorm(x, meanlog = 0, sdlog = sqrt(sig2e_ln)),
```

```
        msmt = dlnorm(x, meanlog = mu_ln, sdlog = sqrt(sig2x_ln +
sig2e_ln)))

## convert to long format
df_ln_long <- df_ln %>%
  pivot_longer(cols = c(signal, me, msmt),
               names_to = 'var', values_to = 'density') %>%
  mutate(var = factor(var, levels = c('me','signal','msmt')))

## create ggplot
p_density_ln <- ggplot(df_ln_long, aes(x = x, y = density, color = var,
linetype = var)) +
  geom_vline(aes(xintercept = 0)) +
  geom_line(aes(size = var)) +
  theme_bw() +
  scale_linetype_manual(
    name = '',
    labels = c(
      'me' = 'Measurement Errors\nLN(0,0.75)',
      'signal' = 'Envr. Signal\nLN(1,1)',
      'msmt' = 'Observed Measurements\nLN(1,1.75)'
    ),
    values = c('me' = 'dashed', 'signal' = 'dashed', 'msmt' = 'solid')
  ) +
  scale_size_manual(
    name = '',
    labels = c(
      'me' = 'Measurement Errors\nLN(0,0.75)',
      'signal' = 'Envr. Signal\nLN(1,1)',
      'msmt' = 'Observed Measurements\nLN(1,1.75)'
    ),
    values = c('me' = 1, 'signal' = 1, 'msmt' = 1.2)
  ) +
  scale_color_manual(
    name = '',
    labels = c(
      'me' = 'Measurement Errors\nLN(0,0.75)',
      'signal' = 'Envr. Signal\nLN(1,1)',
      'msmt' = 'Observed Measurements\nLN(1,1.75)'
    ),
    values = c(
      'me' = 'green',
      'signal' = 'blue',
      'msmt' = 'red'
    )
  ) +
  labs(x = 'Concentration', y = '') +
  scale_x_continuous(limits = c(-1, 20), expand = c(0,0)) +
  scale_y_continuous(limits = c(0, 0.7), expand = c(0,0)) +
  theme(legend.position = 'top',
```

```
        legend.text.align = 0.5,
        axis.text.y = element_blank(),
        axis.ticks.y = element_blank(),
        legend.key.width = unit(1,'cm'),
        panel.grid.minor = element_blank())

p_density_ln

ggsave(filename =
'~/img/ME_manuscript/ggsave_test/Fig_S8_lognormal_density.png',
       plot = p_density_ln, width = 6.5, height = 4, units = 'in', dpi = 300)
```

Code for generating TOC graphic - R

- create separate graphics, save as images
- combine images together in final arrangement

```
## load packages
library(tidyverse)
library(magick)
library(ggtext)
library(viridis)
library(ggdist)

## define colors to use across graphs
pal <- viridis::viridis(3)

me_color  <- pal[3] # measurement errors
sig_color <- pal[2] # environmental signal
obs_color <- pal[1] # observed measurements

# make density graphic ------------------------------------------------

## parameters for distributions
sig2x <- 9 # signal variance
sig2e <- 6 # measurement error variance
mu <- 10   # signal/observed mean

## generate probability density curves
df <- data.frame(x = seq(-10, 30, length.out = 1000))
df <- df %>%
  mutate(signal = dnorm(x, mean = mu, sd = sqrt(sig2x)),
         me = dnorm(x, mean = 0, sd = sqrt(sig2e)),
         msmt = dnorm(x, mean = mu, sd = sqrt(sig2x + sig2e)))

df_long <- df %>%
  pivot_longer(cols = c(signal, me, msmt),
               names_to = 'var', values_to = 'density') %>%
  mutate(var = factor(var, levels = c('me','signal','msmt')))
```

```r
## define legend labels
legend_labels <- c(
  'me' = 'Measurement\nError',
  'signal' = 'Environmental\nSignal',
  'msmt' = 'Observed\nMeasurement'
)

## make plot
p_density <- ggplot(df_long, aes(x = x, y = density,
                                 color = var, linetype = var)) +
  ## add ref line at 0
  geom_vline(aes(xintercept = 0)) +
  ## add density lines
  geom_line(aes(size = var)) +
  ## adjust line types
  scale_linetype_manual(
    name = '', labels = legend_labels,
    values = c('me' = 'dashed', 'signal' = 'dashed', 'msmt' = 'solid')
  ) +
  ## adjust line thickness
  scale_size_manual(
    name = '', labels = legend_labels,
    values = c('me' = 0.8, 'signal' = 0.8, 'msmt' = 1)
  ) +
  ## adjust line colors
  scale_color_manual(
    name = '', labels = legend_labels,
    values = c('me' = me_color, 'signal' = sig_color, 'msmt' = obs_color)
  ) +
  labs(x = 'Concentration',
       title = '',
       #title = NULL
       y = '') +
  scale_x_continuous(limits = c(-8, 27), breaks = seq(-5,25,by=5),
expand=c(0,0)) +
  scale_y_continuous(limits = c(0, 0.2), expand = c(0,0)) +
  ## tweak legend to make thicker lines, move labels
  guides(linetype = guide_legend(override.aes = list(linetype = 'solid',
size=4),
                                 label.position = 'bottom')) +
  theme_bw() +
  theme(
    legend.text = element_text(size=11),
    legend.position = 'bottom',
    legend.text.align = 0.5,
    axis.title.x = element_text(size=11),
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank(),
    legend.key.width = unit(1,'cm'),
```

```
    legend.spacing.x = unit(0.6, 'cm'),
    panel.grid.minor = element_blank(),
    plot.margin = margin(t = 0, r = 0, b = 0, l = 0)
  )

p_density

ggsave(plot = p_density, filename = 'density_viridis.tiff', dpi = 300,
       device = 'tiff', height = 3, width = 4, units = 'in')

# right arrow ------------------------------------------------------------

## coordinates for arrow shape
arrow1_df <- tribble(
  ~x, ~y,
  25, 191,
  25, 129,
  153, 129,
  153, 96,
  272, 160,
  153, 224,
  153, 191,
  25,191
)

## plot
p_arrow <- ggplot(arrow1_df) +
  geom_polygon(aes(x=x, y=y), size=3,
               fill='#4473c4', color = '#2f528f') +
  theme_void() +
  theme(
    plot.background = element_rect(fill = 'white', color = NA),
    plot.margin = margin(t = 0, r = 0, b = 0, l = 0)
  )

p_arrow

ggsave(plot = p_arrow, filename = 'arrow_right.tiff',
       device = 'tiff', dpi = 300)


# corner arrow -----------------------------------------------------------
---

## coordinates for arrow shape
arrow2_df <- tribble(
  ~x, ~y,
  45, 129,
  65, 129,
```

```
  65, 35,
  0, 35,
  0, 15,
  -20, 45,
  0, 75,
  0, 55,
  45, 55,
  45, 129
)

## plot
p_arrow2 <- ggplot(arrow2_df) +
  geom_polygon(aes(x=x, y=y), size=3,
               fill='#4473c4', color = '#2f528f') +
  coord_fixed() +
  theme_void() +
  theme(
    plot.background = element_rect(fill = 'white', color = NA),
    plot.margin = margin(t = 0, r = 0, b = 0, l = 0)
  )

p_arrow2
ggsave(plot = p_arrow2, filename = 'arrow_corner.tiff',
       device = 'tiff', dpi = 300)

# ANOVA equation  ------------------------------------------------------

## load Cambria Math font
windowsFonts(cm = windowsFont('Cambria Math'))

## info for equation - greek letters and symbols
greek_df <- data.frame(
  x = c(-0.025, -0.009, 0.005, 0.02,
        -0.016,-0.001, 0.013), y = 0,
  label = c('<i>Y<sub>ij</sub></i>','<i>\U03BC</i>',
            '<i>\U03C4<sub>i</sub></i>','<i>\U03B5<sub>ij</sub></i>',
            '&#x3D;','\U002B','\U002B'),
  color = c(obs_color, sig_color, sig_color, me_color,
'black','black','black')
)

## info for brackets beneath equation
brace_df <- data.frame(
  x = c(-0.025, -0.009, 0.005, 0.02), y = -0.12,
  color = c(pal[1], pal[2], pal[2], pal[3])
)

## info for descriptions beneath brackets
desc_df <- data.frame(
```

```
  x = c(-0.025, -0.009, 0.006, 0.021), y = -0.23,
  label = c('Observed\nMeasurement','True\nConcentration',
            'Site-level\nVariation','Measurement\nError')
)


## plot
p_eqn <- ggplot() +
  ## add braces
  geom_text(data = brace_df, aes(x, y, color=color),
            size=12, angle = 90, label = '{', hjust = 0.5) +
  ## add text descriptions below braces
  geom_text(data = desc_df, aes(x,y, label = label),
            hjust = 0.5, size = rel(3.5)) +
  ## add equation symbols
  ggtext::geom_richtext(data = greek_df,
                        aes(x=x, y=y,
                            #color = color,
                            label = label),
                        hjust = 0.5, size = 12,  family = 'cm',
                        fill = NA, label.color = NA) +
  ## add caption
  annotate('text', x=0.015, y=-0.32, label='Sites = i, replicates = j') +
  scale_color_identity() +
  scale_x_continuous(limits=c(-0.029, 0.028)) +
  scale_y_continuous(limits=c(-0.35,0.15)) +
  labs(caption = NULL, title = NULL) +
  theme_void() +
  theme(
    plot.background = element_rect(color = NA, fill = 'white'),
    plot.margin = margin(t = 0, r = 0, b = 0, l = 0)
  )

p_eqn

ggsave(plot = p_eqn, filename = 'equation_viridis.tiff', dpi = 300,
       units = 'in', height = 2.45, width = 4)

# Confidence intervals ----------------------------------------------------

p_conf <-  ggplot() +
  ## rectangles for Measurement error region
  geom_interval(aes(y = 3, x = 4.5, xmin = 3.5, xmax=3.75),
                color = me_color, interval_size_range = c(1,16)) +
  geom_interval(aes(y = 3, x = 4.5, xmin = 5.25, xmax=5.5),
                color = me_color, interval_size_range = c(1,16)) +
  ## rectangles for signal region
  geom_interval(aes(y = 3, x = 4.5, xmin = 3.75, xmax=5.25),
                color = sig_color, interval_size_range = c(1,16)) +
```

```
  ## CI for Observed Mean
  annotate('errorbarh', y = 3, xmin=3.5, xmax=5.5, height=0.2, color =
obs_color,
           size=1.2) +
  annotate('point', x=4.5, y=3, fill= obs_color, size=8, shape=21) +
  ## CI for signal mean
  annotate('errorbarh', y = 2.2, xmin = 3.75, xmax=5.25, height=0.2, color =
sig_color,
           size=1.2) +
  annotate('point', x=4.5, y=2.2, fill= sig_color, size=8, shape = 22) +
  ## reference line and label for mu
  annotate('segment', x=4.35, xend=4.35, y=1.8, yend=3.6, linetype =
'dashed', size=1.2) +
  ggtext::geom_richtext(aes(x=4.35, y=1.65), label='&mu;',
                        label.color = NA, size=8, family='cm') +
  #labs(title = '') +
  labs(title = NULL) +
  scale_y_continuous(limits = c(1.6,3.8), breaks = c(2.2,3),
                     labels = c('Signal Mean',
                                'Observed Mean'),
                     position = 'left') +
  theme(
    axis.title = element_blank(),
    axis.ticks = element_blank(),
    axis.text.x = element_blank(),
    axis.text.y = element_text(size=rel(2.5)),
    panel.grid = element_blank(),
    plot.background = element_rect(fill = 'white', color = NA),
    panel.background = element_rect(fill = 'white', color = NA),
    panel.border = element_blank(),
    plot.margin = margin(t = 0, r = 0, b = 0, l = 0)
  )

p_conf

ggsave(plot = p_conf, filename = 'ci_plot_viridis.tiff', dpi = 300,
       units = 'in', height = 3, width = 8, device = 'tiff')


# create blank slivers ----------------------------------------------------

## tall & skinny, goes between density plot and right arrow
tiff("blank1.tiff", width = 40, height = 300, units="px", res=300)
dev.off()

## goes above right arrow
tiff("blank2.tiff", width = 184, height = 100, units="px", res=300)
dev.off()
```

```
## goes to left of CI plot
tiff('blank3.tiff', width = 200, height = 164, units='px', res = 300)
dev.off()

## short and wide, goes between 2 rows
tiff('blank4.tiff', width = 800, height = 50, units='px', res = 300)
dev.off()

# combine graphics with magick package ----------------------------------

## read blank images
img_blank1 <- image_read('blank1.tiff')
img_blank2 <- image_read('blank2.tiff')
img_blank3 <- image_read('blank3.tiff')
img_blank4 <- image_read('blank4.tiff')

## read and resize density image, add title
img_density <- image_read('density_viridis.tiff') %>%
  image_resize('x500') %>%
  image_annotate(text = 'Data Distribution',
                 location = '+210+0',
                 #location = '+200+20',
                 size=34)

## read and resize right arrow image
img_arrow <- image_read('arrow_right.tiff') %>%
  image_resize('x100')

## read and resize corner arrow iamge
img_arrow3 <- image_read('arrow_corner.tiff') %>%
  image_resize('x200')

## read and resize equation image, add title
img_eqn <- image_read('equation_viridis.tiff') %>%
  image_resize('x500') %>%
  image_crop(geometry = 'x450+0+50') %>%
  image_annotate(text = 'ANOVA Equation',
                 #location = '+150+20',
                 location = '+250+0',
                 size=34)

## read and resize CI image, add title
img_conf <- image_read('ci_plot_viridis.tiff') %>%
  image_resize('x300') %>%
  image_annotate(text = '95% Confidence Intervals',
                 #location = '+280+20',
                 location = '+285+0',
                 size=34)
```

```
## surround arrow with blank space
img_arrow <- image_append(c(img_blank2, img_arrow), stack = TRUE)
img_arrow <- image_append(c(img_blank1, img_arrow))

img_arrow <- img_arrow %>%
  image_crop(geometry = '200x300+0+0')

## arrange in 2 rows, append for final graphic
img_row1 <- image_append(c(img_density, img_arrow, img_eqn))
img_row2 <- image_append(c(img_blank3, img_blank3, img_blank3, img_conf,
img_arrow3))

img_final <- image_append(c(img_row1, img_blank4, img_row2), stack = TRUE)
img_final

## convert final graphic to output size
img_final %>%
  image_scale(geometry = '975x525') %>%
  image_write(path = 'TOC_graphic_R.tiff', format = 'tiff',
              density = 300)

## crop edges of image and resize
image_read('TOC_graphic_R.tiff') %>%
  image_crop(geometry = '945x525+12+0') %>%
  image_scale(geometry = '975x525') %>%
  image_write(path = 'TOC_graphic_cropped.tiff', format = 'tiff',
              density = 300)
```