prelim_sims

Setup for simulation

```r
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```r
library(ggplot2)
```

```
Warning: package 'ggplot2' was built under R version 4.3.3
```

```r
library(gt)
```

```
Warning: package 'gt' was built under R version 4.3.3
```

```r
library(knitr)
```

```
Warning: package 'knitr' was built under R version 4.3.3
```

```r
library(lavaan)
```

```
Warning: package 'lavaan' was built under R version 4.3.3

This is lavaan 0.6-18
lavaan is FREE software! Please report any bugs.
```

```r
library(pinsearch)
library(SimDesign)
```

```
Warning: package 'SimDesign' was built under R version 4.3.3
```

```r
# TODO:
#   - Summarize the pattern of bias

# Define conditions: Testing different sample sizes
design <- createDesign(
  n = c(30, 100, 250, 1000, 5000)
)

# Fixed objects
set.seed(1855)

# Helper Function
# Generates a random covariance matrix
get_ucov <- function(p, scale = sqrt(0.1), n = 5) {
  W <- matrix(rnorm(p * n), nrow = n)
  WtW <- crossprod(W)
  D <- diag(1 / sqrt(diag(WtW))) * scale
  D %*% WtW %*% D
}
# Defines key parameters
fixed <- list(
  p = 6,
  lambda = c(.3, .7, .4, .5, .6, .4),
  dlambda = list(
    c(0, 0, 0, 0, 0, 0),
    c(.1, 0, 0, 0, 0, 0),
    c(.2, -.3, 0, 0, 0, 0),
```

```r
    c(.3, -.3, 0, 0, 0, 0)
  ),
  nu = c(2, 3, 1.5, 3.5, 2, 3),
  alpha = c(0, -0.25, 0.25, 0.5),
  psi = c(1, 0.85, 1.15, 0.7),
  theta = c(1, 1.2, .8, .9, 1, 1) - .1,
  dtheta = matrix(
    runif(24, min = -0.2, max = 0.2),
    nrow = 4
  ),
  # ucov = replicate(4, get_ucov(6), simplify = FALSE)
  ucov = replicate(4, diag(.1, 6), simplify = FALSE),
  ninv_ind = c(1, 2)
)
# lavaan syntax for the measurement model
fixed$mod <- paste(
  "f =~",
  paste0("y", seq_len(fixed$p), collapse = " + ")
)
# Compute implied means and covariances
fixed <- within(fixed, {
  lambdag <- lapply(dlambda, FUN = \(x) x + lambda)
  Thetag <- lapply(seq_along(ucov),
                   FUN = function(g) {
                     diag(theta + dtheta[g, ]) + ucov[[g]]
                   })
  covy <- mapply(\(lam, psi, th) tcrossprod(lam) * psi + th,
                 lam = lambdag, psi = psi, th = Thetag,
                 SIMPLIFY = FALSE)
  meany <- mapply(\(lam, al, nu) nu + lam * al,
                  lam = lambdag, al = alpha, nu = list(nu),
                  SIMPLIFY = FALSE)
})

# Population effect size
fixed$fmacs_pop <- local({
  pooled_sd <- lapply(fixed$covy, FUN = \(x) diag(x)) |>
    do.call(what = rbind) |>
    colMeans() |>
    sqrt()
```

```r
  fmacs(
    intercepts = matrix(rep(fixed$nu, 4),
                        nrow = 4,
                        byrow = TRUE
    ),
    loadings = sweep(
      do.call(rbind, fixed$dlambda),
      MARGIN = 2,
      STATS = fixed$lambda,
      FUN = "+"
    ),
    latent_mean = 0,
    latent_sd = 1,
    pooled_item_sd = pooled_sd
  )[1:2]
})

# Function for data generation
# sim_y <- function(n, lambda, nu, alpha, psi, Theta) {
#     covy <- tcrossprod(lambda) * psi + Theta
#     meany <- nu + lambda * alpha
#     MASS::mvrnorm(n, mu = meany, Sigma = covy)
# }
```

Running the simulation

```r
generate <- function(condition, fixed_objects) {
  ylist <- lapply(seq_along(fixed_objects$covy),
                  FUN = function(g) {
                    yg <- MASS::mvrnorm(
                      condition$n,
                      mu = fixed_objects$meany[[g]],
                      Sigma = fixed_objects$covy[[g]]
                    )
                    colnames(yg) <- paste0("y", seq_len(fixed_objects$p))
                    cbind(yg, group  = g)
                  })
  do.call(rbind, ylist)
}
sim1 <- generate(design[3, ], fixed_objects = fixed)
```

```r
# Evaluate/Summarize
evaluate <- function(condition, results, fixed_objects) {
  c(
    bias = colMeans(results) - fixed_objects$fmacs_pop,
    robust_bias = apply(results, 2, mean, trim = .1) -
      fixed_objects$fmacs_pop,
    emp_sd = apply(results, 2, sd),
    emp_mad = apply(results, 2, mad)
  )
}

# Analysis
analyze <- function(condition, dat, fixed_objects) {
  # Define lavaan syntax
  pinv_fit <- cfa(
    fixed_objects$mod,
    data = dat,
    group = "group", std.lv = TRUE,
    group.equal = c("loadings", "intercepts"),
    group.partial = c(
      paste0("f=~y", fixed_objects$ninv_ind),
      paste0("y", fixed_objects$ninv_ind, "~1")
    )
  )
  as.vector(pinsearch::pin_effsize(pinv_fit))
}
```

```r
out <- runSimulation(design,
                     replications = 500,
                     parallel = TRUE,
                     generate = generate,
                     analyse = analyze,
                     summarise = evaluate,
                     filename = "results-trial1",
                     packages = c("MASS", "lavaan", "pinsearch"),
                     fixed_objects = fixed
)
```

```
Number of parallel clusters in use: 19
```

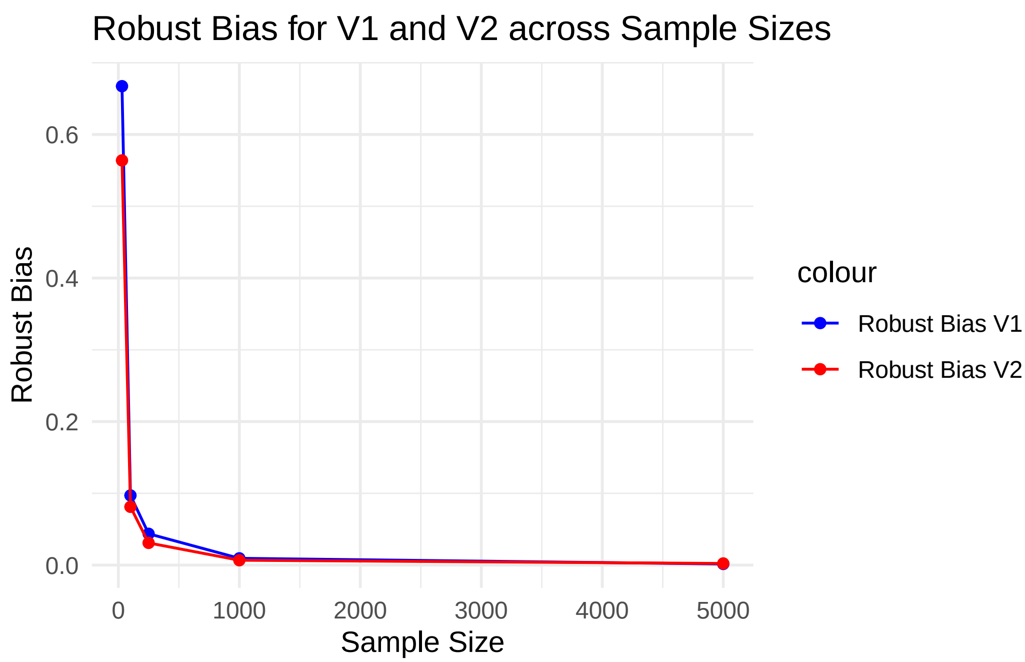
```
Design: 1/5;   RAM Used: 71.7 Mb;   Replications: 500;   Total Time: 0.00s
 Conditions: n=30


Design: 2/5;   RAM Used: 72.5 Mb;   Replications: 500;   Total Time: 03m 28.72s
 Conditions: n=100


Design: 3/5;   RAM Used: 72.5 Mb;   Replications: 500;   Total Time: 03m 52.28s
 Conditions: n=250


Design: 4/5;   RAM Used: 72.5 Mb;   Replications: 500;   Total Time: 04m 4.67s
 Conditions: n=1000


Design: 5/5;   RAM Used: 72.5 Mb;   Replications: 500;   Total Time: 04m 18.74s
 Conditions: n=5000


Simulation complete. Total execution time: 04m 41.61s


Saving simulation results to file: results-trial1.rds
```

```r
data <- readRDS("results-trial1.rds")
```

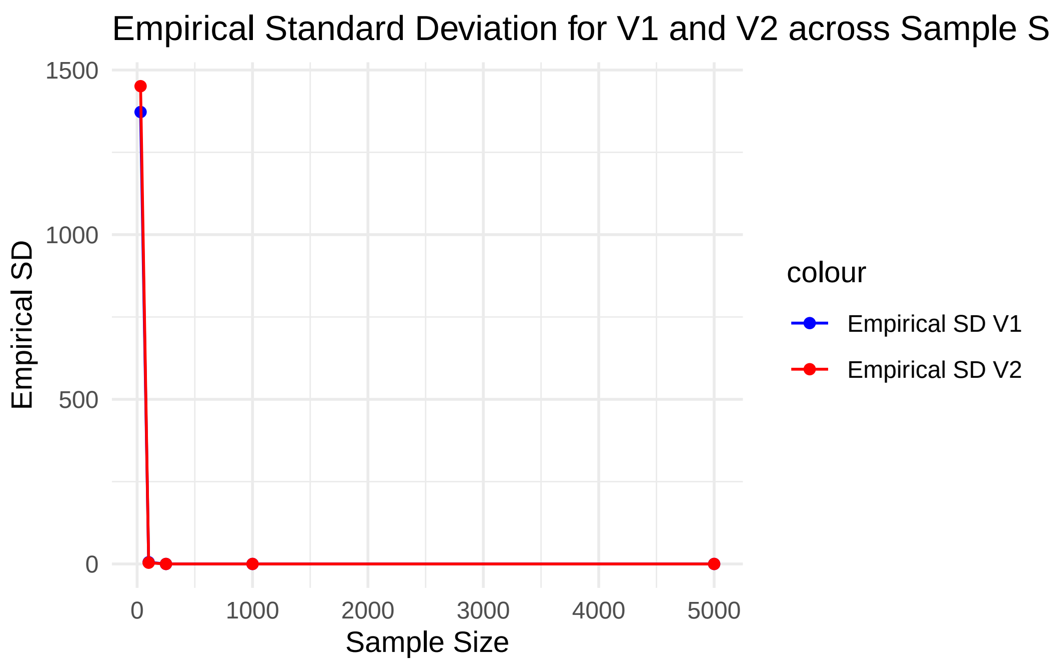
Graphs to visualize simulation results

```r
ggplot(out, aes(x = n)) +
  geom_line(aes(y = bias.V1, color = "Bias V1")) +
  geom_point(aes(y = bias.V1, color = "Bias V1")) +
  geom_line(aes(y = bias.V2, color = "Bias V2")) +
  geom_point(aes(y = bias.V2, color = "Bias V2")) +
  labs(title = "Bias for V1 and V2 across Sample Sizes",
       x = "Sample Size", y = "Bias") +
  scale_color_manual(values = c("Bias V1" = "blue", "Bias V2" = "red")) +
  theme_minimal()
```

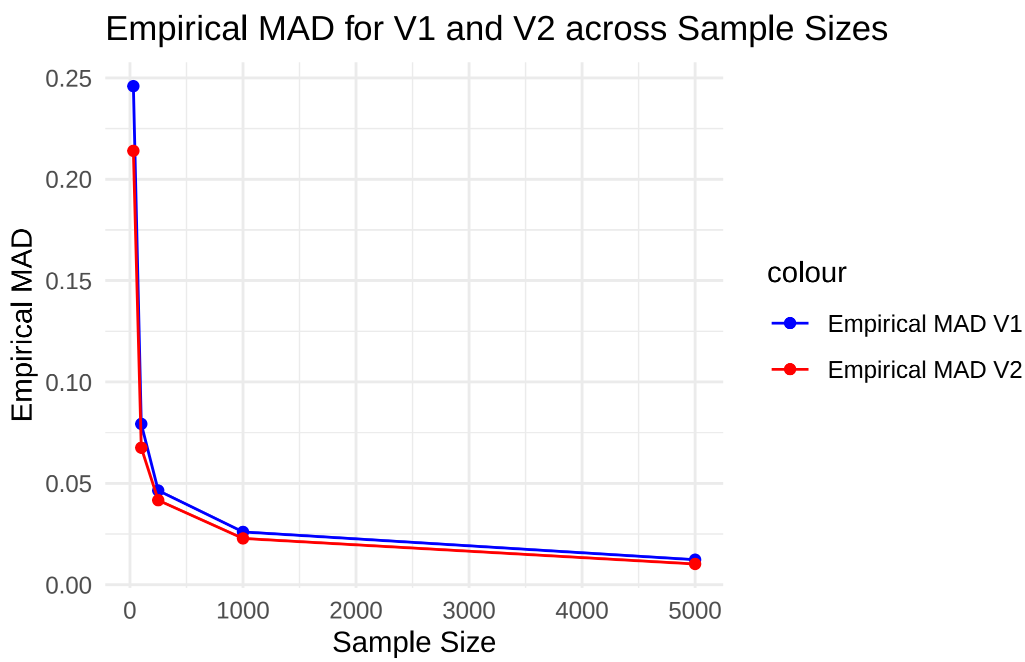

```r
# Robust Bias for V1 and V2
ggplot(out, aes(x = n)) +
  geom_line(aes(y = robust_bias.V1, color = "Robust Bias V1")) +
  geom_point(aes(y = robust_bias.V1, color = "Robust Bias V1")) +
  geom_line(aes(y = robust_bias.V2, color = "Robust Bias V2")) +
  geom_point(aes(y = robust_bias.V2, color = "Robust Bias V2")) +
  labs(title = "Robust Bias for V1 and V2 across Sample Sizes",
       x = "Sample Size", y = "Robust Bias") +
  scale_color_manual(values = c("Robust Bias V1" = "blue", "Robust Bias V2" = "red")) +
  theme_minimal()
```




```
# Empirical Standard Deviation for V1 and V2
ggplot(out, aes(x = n)) +
  geom_line(aes(y = emp_sd.V1, color = "Empirical SD V1")) +
  geom_point(aes(y = emp_sd.V1, color = "Empirical SD V1")) +
  geom_line(aes(y = emp_sd.V2, color = "Empirical SD V2")) +
  geom_point(aes(y = emp_sd.V2, color = "Empirical SD V2")) +
  labs(title = "Empirical Standard Deviation for V1 and V2 across Sample Sizes",
       x = "Sample Size", y = "Empirical SD") +
  scale_color_manual(values = c("Empirical SD V1" = "blue", "Empirical SD V2" = "red")) +
  theme_minimal()
```




```r
# Empirical MAD for V1 and V2
ggplot(out, aes(x = n)) +
  geom_line(aes(y = emp_mad.V1, color = "Empirical MAD V1")) +
  geom_point(aes(y = emp_mad.V1, color = "Empirical MAD V1")) +
  geom_line(aes(y = emp_mad.V2, color = "Empirical MAD V2")) +
  geom_point(aes(y = emp_mad.V2, color = "Empirical MAD V2")) +
  labs(title = "Empirical MAD for V1 and V2 across Sample Sizes",
       x = "Sample Size", y = "Empirical MAD") +
  scale_color_manual(values = c("Empirical MAD V1" = "blue", "Empirical MAD V2" = "red"))
  theme_minimal()
```



Set up for tables

```
bias_table <- data.frame(
  SampleSize = out$n,
  Bias_V1 = out$bias.V1,
  Bias_V2 = out$bias.V2
)

robust_bias_table <- data.frame(
  SampleSize = out$n,
  Robust_Bias_V1 = out$robust_bias.V1,
  Robust_Bias_V2 = out$robust_bias.V2
)

emp_sd_table <- data.frame(
  SampleSize = out$n,
  Empirical_SD_V1 = out$emp_sd.V1,
  Empirical_SD_V2 = out$emp_sd.V2
)

emp_mad_table <- data.frame(
  SampleSize = out$n,
```

```
  Empirical_MAD_V1 = out$emp_mad.V1,
  Empirical_MAD_V2 = out$emp_mad.V2
)
```

```
# Inline tables
knitr::kable(bias_table, caption = "Bias for V1 and V2 across Sample Sizes")
```

Table 1: Bias for V1 and V2 across Sample Sizes

SampleSize	Bias_V1	Bias_V2
30	180.6725954	168.0517972
100	0.3518730	0.2667248
250	0.0478055	0.0317518
1000	0.0105088	0.0070487
5000	0.0016523	0.0022065

```
knitr::kable(robust_bias_table, caption = "Robust Bias for V1 and V2 across Sample Sizes")
```

Table 2: Robust Bias for V1 and V2 across Sample Sizes

SampleSize	Robust_Bias_V1	Robust_Bias_V2
30	0.6673828	0.5639481
100	0.0970547	0.0811938
250	0.0436491	0.0309130
1000	0.0094398	0.0067654
5000	0.0015737	0.0022893

```
knitr::kable(emp_sd_table, caption = "Empirical Standard Deviation for V1 and V2 across Sa
```

Table 3: Empirical Standard Deviation for V1 and V2 across Sample Sizes

SampleSize	Empirical_SD_V1	Empirical_SD_V2
30	1372.4076599	1450.8974411
100	5.2639797	3.8151992
250	0.0529399	0.0424741
1000	0.0263212	0.0244311
5000	0.0122347	0.0104357

```r
knitr::kable(emp_mad_table, caption = "Empirical MAD for V1 and V2 across Sample Sizes")
```

Table 4: Empirical MAD for V1 and V2 across Sample Sizes

SampleSize	Empirical_MAD_V1	Empirical_MAD_V2
30	0.2459071	0.2140147
100	0.0793031	0.0675686
250	0.0464374	0.0416447
1000	0.0260422	0.0228123
5000	0.0123308	0.0102229