CFA_SimDesign

```r
library(ggplot2)
library(lavaan)
library(parallel)
library(SimDesign)
library(tidyr)
```

Step 1 — Defining conditions (sample size and model)

```r
Design <- createDesign(sample_size = c(100, 200, 400, 800, 1200),
                       model_type = c("true", "misspecified"),
                       data_type = c("normal", "non-normal"))
```

Step 2 — Define generate, analyse, and summarize functions

```r
# Generate function: Simulate data based on the true model
Generate <- function(condition, fixed_objects) {
    N <- condition$sample_size

    # True model: Two factors, 8 indicators each
    true_model <- '
        f1 =~ 0.7*y1 +
        0.7*y2 +
        0.7*y3 +
        0.7*y4 +
        0.7*y5 +
        0.7*y6 +
        0.7*y7 +
        0.7*y8
```

```
    f2 =~ 0.7*y9 +
    0.7*y10 +
    0.7*y11 +
    0.7*y12 +
    0.7*y13 +
    0.7*y14 +
    0.7*y15 +
    0.7*y16
'


# Misspecification: Add cross-loadings
misspecified_model <- '
    f1 =~ 0.7*y1 +
    0.7*y2 +
    0.7*y3 +
    0.7*y4 +
    0.7*y5 +
    0.7*y6 +
    0.7*y7 +
    0.7*y8 +
    0.3*y9 # added cross-loading

    f2 =~ 0.7*y9 +
    0.7*y10 +
    0.7*y11 +
    0.7*y12 +
    0.7*y13 +
    0.7*y14 +
    0.7*y15 +
    0.7*y16 +
    0.3*y1 # added cross-loading
'

# Generate data based on the condition
model_to_use <- ifelse(condition$model_type == "true", true_model, misspecified_model)

if (condition$data_type == "normal") {
  data <- lavaan::simulateData(model_to_use, sample.nobs = N)
} else {
  data <- lavaan::simulateData(model_to_use, sample.nobs = N, skewness = 5, kurtosis = 2
```

```
  }

    return(data)
}


# Analyse function: Fit the true model and extract fit measures
Analyse <- function(condition, dat, fixed_objects) {
    true_model <- '
        f1 =~ y1 + y2 + y3 + y4 + y5 + y6 + y7 + y8
        f2 =~ y9 + y10 + y11 + y12 + y13 + y14 + y15 + y16
    '

    fit <- lavaan::cfa(true_model, data = dat, estimator = "ML")

    # Extract fit measures
    fit_measures <- lavaan::fitMeasures(fit, c("cfi", "tli", "rmsea", "srmr", "chisq", "df
    return(fit_measures)
}


# Summarise function: Calculate the average fit measures from n replications
Summarise <- function(condition, results, fixed_objects) {
    summary_measures <- colMeans(results)
    return(summary_measures)
}
```

Step 3 Collect results by looping over the rows in design

```
# Make sure it doesn't try resuming a simulation
SimClean()


res <- SimDesign::runSimulation(design=Design, replications=1000,
                      generate=Generate, analyse=Analyse, summarise=Summarise,
                      parallel = TRUE)
```

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

```
Design: 1/20;   RAM Used: 69.7 Mb;   Replications: 1000;   Total Time: 0.00s
 Conditions: sample_size=100, model_type=true, data_type=normal


Design: 2/20;   RAM Used: 70.6 Mb;   Replications: 1000;   Total Time: 8.53s
 Conditions: sample_size=200, model_type=true, data_type=normal


Design: 3/20;   RAM Used: 70.6 Mb;   Replications: 1000;   Total Time: 16.32s
 Conditions: sample_size=400, model_type=true, data_type=normal


Design: 4/20;   RAM Used: 70.7 Mb;   Replications: 1000;   Total Time: 24.21s
 Conditions: sample_size=800, model_type=true, data_type=normal


Design: 5/20;   RAM Used: 70.7 Mb;   Replications: 1000;   Total Time: 32.09s
 Conditions: sample_size=1200, model_type=true, data_type=normal


Design: 6/20;   RAM Used: 70.8 Mb;   Replications: 1000;   Total Time: 40.06s
 Conditions: sample_size=100, model_type=misspecified, data_type=normal


Design: 7/20;   RAM Used: 70.8 Mb;   Replications: 1000;   Total Time: 47.88s
 Conditions: sample_size=200, model_type=misspecified, data_type=normal


Design: 8/20;   RAM Used: 70.9 Mb;   Replications: 1000;   Total Time: 55.65s
 Conditions: sample_size=400, model_type=misspecified, data_type=normal


Design: 9/20;   RAM Used: 70.9 Mb;   Replications: 1000;   Total Time: 01m 3.49s
 Conditions: sample_size=800, model_type=misspecified, data_type=normal


Design: 10/20;   RAM Used: 71 Mb;   Replications: 1000;   Total Time: 01m 11.35s
 Conditions: sample_size=1200, model_type=misspecified, data_type=normal


Design: 11/20;   RAM Used: 71 Mb;   Replications: 1000;   Total Time: 01m 19.41s
```

```
 Conditions: sample_size=100, model_type=true, data_type=non-normal


Design: 12/20;   RAM Used: 71.1 Mb;   Replications: 1000;   Total Time: 01m 30.16s
 Conditions: sample_size=200, model_type=true, data_type=non-normal


Design: 13/20;   RAM Used: 71.1 Mb;   Replications: 1000;   Total Time: 01m 40.58s
 Conditions: sample_size=400, model_type=true, data_type=non-normal


Design: 14/20;   RAM Used: 71.2 Mb;   Replications: 1000;   Total Time: 01m 50.80s
 Conditions: sample_size=800, model_type=true, data_type=non-normal


Design: 15/20;   RAM Used: 71.2 Mb;   Replications: 1000;   Total Time: 02m 1.10s
 Conditions: sample_size=1200, model_type=true, data_type=non-normal


Design: 16/20;   RAM Used: 71.3 Mb;   Replications: 1000;   Total Time: 02m 11.57s
 Conditions: sample_size=100, model_type=misspecified, data_type=non-normal


Design: 17/20;   RAM Used: 71.3 Mb;   Replications: 1000;   Total Time: 02m 22.82s
 Conditions: sample_size=200, model_type=misspecified, data_type=non-normal


Design: 18/20;   RAM Used: 71.4 Mb;   Replications: 1000;   Total Time: 02m 33.29s
 Conditions: sample_size=400, model_type=misspecified, data_type=non-normal


Design: 19/20;   RAM Used: 71.4 Mb;   Replications: 1000;   Total Time: 02m 43.70s
 Conditions: sample_size=800, model_type=misspecified, data_type=non-normal


Design: 20/20;   RAM Used: 71.5 Mb;   Replications: 1000;   Total Time: 02m 54.26s
 Conditions: sample_size=1200, model_type=misspecified, data_type=non-normal


Simulation complete. Total execution time: 03m 4.81s
```

```r
res <- res %>% dplyr::select(-COMPLETED, -SEED, -REPLICATIONS, RAM_USED)

# ------------------------------------------------------
# ------------------------------------------------------
print(res, n = 100)
```

```
# A tibble: 20 x 13
   sample_size model_type   data_type      cfi     tli     rmsea     srmr  chisq
         <dbl> <chr>        <chr>        <dbl>   <dbl>     <dbl>    <dbl>  <dbl>
 1         100 true         normal     0.96627 0.97096 0.025780  0.072424 111.72
 2         200 true         normal     0.98680 0.99167 0.014805  0.051260 107.94
 3         400 true         normal     0.99510 0.99906 0.0081434 0.035831 104.27
 4         800 true         normal     0.99771 0.99985 0.0054859 0.025423 103.41
 5        1200 true         normal     0.99843 0.99990 0.0044723 0.020798 103.37
 6         100 misspecified normal     0.93732 0.92974 0.040974  0.084181 124.05
 7         200 misspecified normal     0.95699 0.95056 0.034750  0.066860 131.38
 8         400 misspecified normal     0.96143 0.95507 0.034383  0.056558 153.83
 9         800 misspecified normal     0.96319 0.95712 0.034012  0.050354 199.80
10        1200 misspecified normal     0.96310 0.95701 0.034148  0.048391 248.53
11         100 true         non-normal 0.85760 0.83949 0.048941  0.082828 132.45
12         200 true         non-normal 0.92406 0.91437 0.033001  0.059129 129.96
13         400 true         non-normal 0.96209 0.95717 0.022437  0.041808 127.79
14         800 true         non-normal 0.97979 0.97695 0.016286  0.029830 128.70
15        1200 true         non-normal 0.98653 0.98464 0.013173  0.024361 128.45
16         100 misspecified non-normal 0.81637 0.78833 0.058338  0.088842 142.10
17         200 misspecified non-normal 0.88751 0.86952 0.042949  0.066343 144.64
18         400 misspecified non-normal 0.92300 0.91039 0.034861  0.051337 156.17
19         800 misspecified non-normal 0.94277 0.93333 0.029928  0.041672 179.10
20        1200 misspecified non-normal 0.95047 0.94229 0.027813  0.037731 200.63
# i 5 more variables: df <dbl>, SIM_TIME <chr>, RAM_USED <chr>, ERRORS <int>,
#   WARNINGS <int>
```

```r
true_model <- '
        f1 =~ 0.7*y1 +
        0.7*y2 +
        0.7*y3 +
        0.7*y4 +
        0.7*y5 +
        0.7*y6 +
        0.7*y7 +
        0.7*y8
```

```
        f2 =~ 0.7*y9 +
        0.7*y10 +
        0.7*y11 +
        0.7*y12 +
        0.7*y13 +
        0.7*y14 +
        0.7*y15 +
        0.7*y16
    '

system.time({
  simulated_true_data <- simulateData(true_model, sample.nobs = 1e6, skewness = 0, kurtosi

  fit <- lavaan::cfa(true_model, data = simulated_true_data)
})
```

```
   user  system elapsed
   0.61    0.09    2.40
```

```
summary(fit, fit.measures = TRUE)
```

```
lavaan 0.6-18 ended normally after 19 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        19

  Number of observations                       1000000

Model Test User Model:

  Test statistic                               118.929
  Degrees of freedom                               117
  P-value (Chi-square)                           0.433

Model Test Baseline Model:

  Test statistic                           3195832.045
  Degrees of freedom                               120
  P-value                                        0.000
```

```
User Model versus Baseline Model:

  Comparative Fit Index (CFI)                    1.000
  Tucker-Lewis Index (TLI)                       1.000

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)          -24296521.603
  Loglikelihood unrestricted model (H1)  -24296462.138

  Akaike (AIC)                            48593081.206
  Bayesian (BIC)                          48593305.700
  Sample-size adjusted Bayesian (SABIC)   48593245.317

Root Mean Square Error of Approximation:

  RMSEA                                          0.000
  90 Percent confidence interval - lower         0.000
  90 Percent confidence interval - upper         0.001
  P-value H_0: RMSEA <= 0.050                    1.000
  P-value H_0: RMSEA >= 0.080                    0.000

Standardized Root Mean Square Residual:

  SRMR                                           0.001

Parameter Estimates:

  Standard errors                             Standard
  Information                                 Expected
  Information saturated (h1) model          Structured

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  f1 =~
    y1                0.700
    y2                0.700
    y3                0.700
    y4                0.700
    y5                0.700
    y6                0.700
    y7                0.700
```

```
    y8                0.700
  f2 =~
    y9                0.700
    y10               0.700
    y11               0.700
    y12               0.700
    y13               0.700
    y14               0.700
    y15               0.700
    y16               0.700

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)
  f1 ~~
    f2               -0.001    0.001   -1.110    0.267

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1                1.001    0.002  636.241    0.000
   .y2                1.000    0.002  636.174    0.000
   .y3                0.999    0.002  636.109    0.000
   .y4                0.998    0.002  636.067    0.000
   .y5                1.000    0.002  636.209    0.000
   .y6                1.002    0.002  636.302    0.000
   .y7                0.998    0.002  636.044    0.000
   .y8                1.000    0.002  636.192    0.000
   .y9                1.002    0.002  636.228    0.000
   .y10               1.000    0.002  636.139    0.000
   .y11               0.998    0.002  636.002    0.000
   .y12               1.001    0.002  636.167    0.000
   .y13               0.999    0.002  636.050    0.000
   .y14               1.000    0.002  636.120    0.000
   .y15               0.999    0.002  636.066    0.000
   .y16               1.002    0.002  636.282    0.000
    f1                0.999    0.002  561.647    0.000
    f2                1.002    0.002  561.937    0.000
```

```
library(dplyr)
```

```
Attaching package: 'dplyr'
```

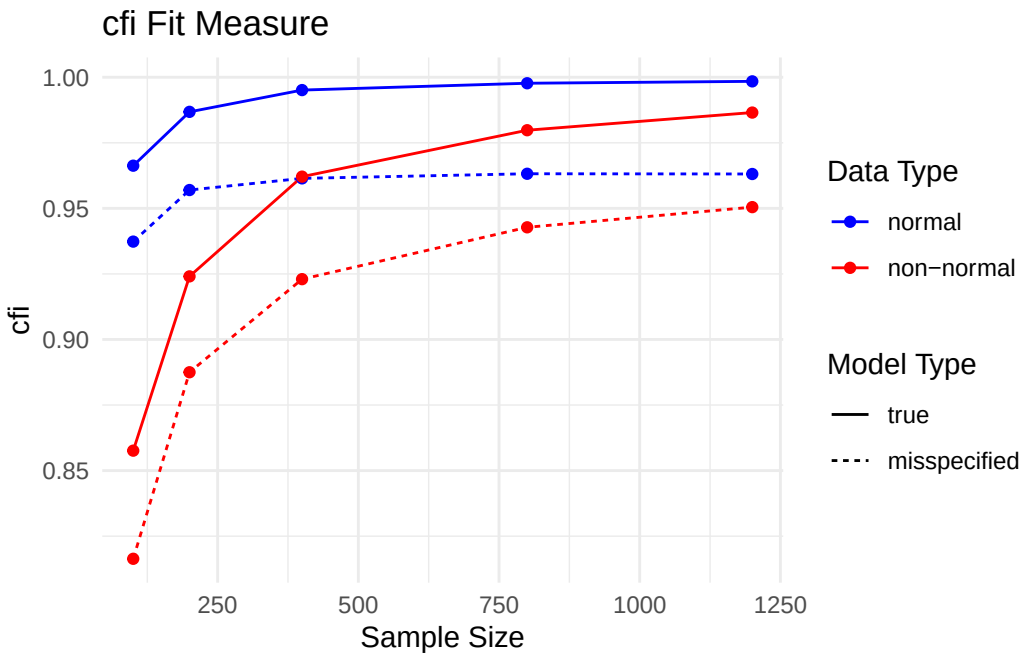
```
The following objects are masked from 'package:stats':

    filter, lag

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

    intersect, setdiff, setequal, union
```

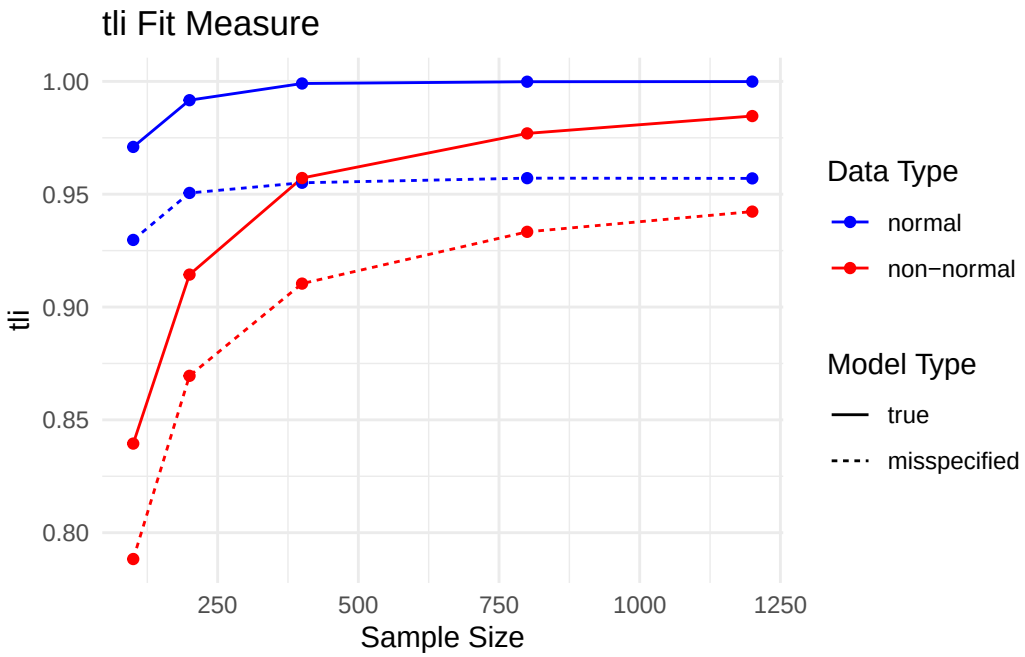
```r
library(ggplot2)

res_long <- res %>%
    pivot_longer(cols = starts_with("cfi") | starts_with("tli") | starts_with("rmsea") | s
                 names_to = "fit_measure",
                 values_to = "value") %>%
    separate(fit_measure, into = c("measure", "type"), sep = "_") %>%
    mutate(type = factor(model_type, levels = c("true", "misspecified")),
           data_type = factor(data_type, levels = c("normal", "non-normal")),
           measure = factor(measure, levels = c("cfi", "tli", "rmsea", "srmr", "chisq")))
```

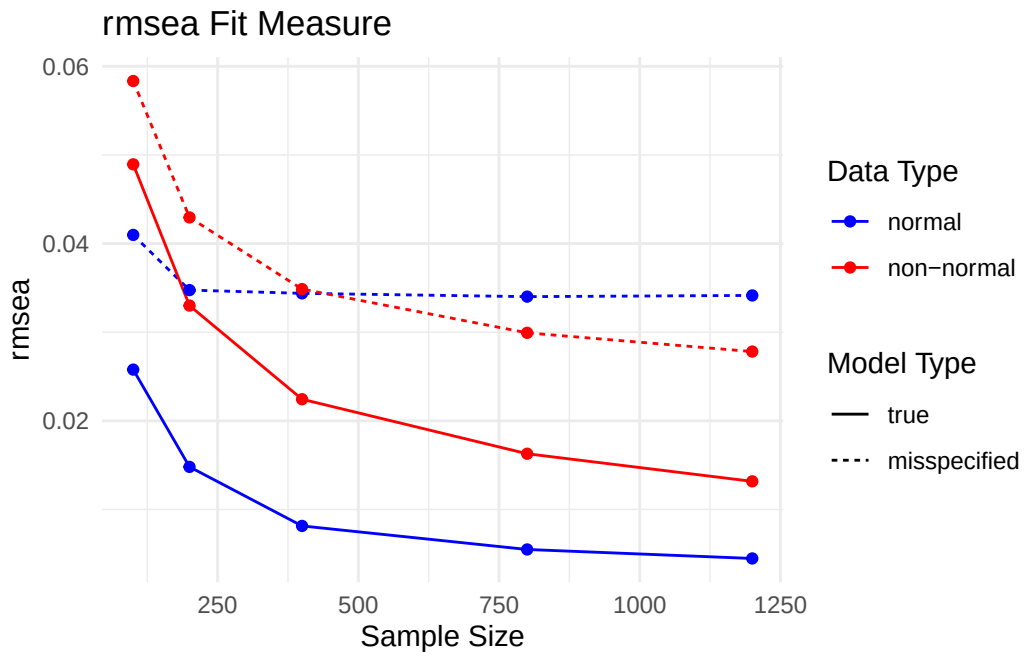
```
Warning: Expected 2 pieces. Missing pieces filled with `NA` in 100 rows [1, 2, 3, 4, 5,
6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, ...].
```

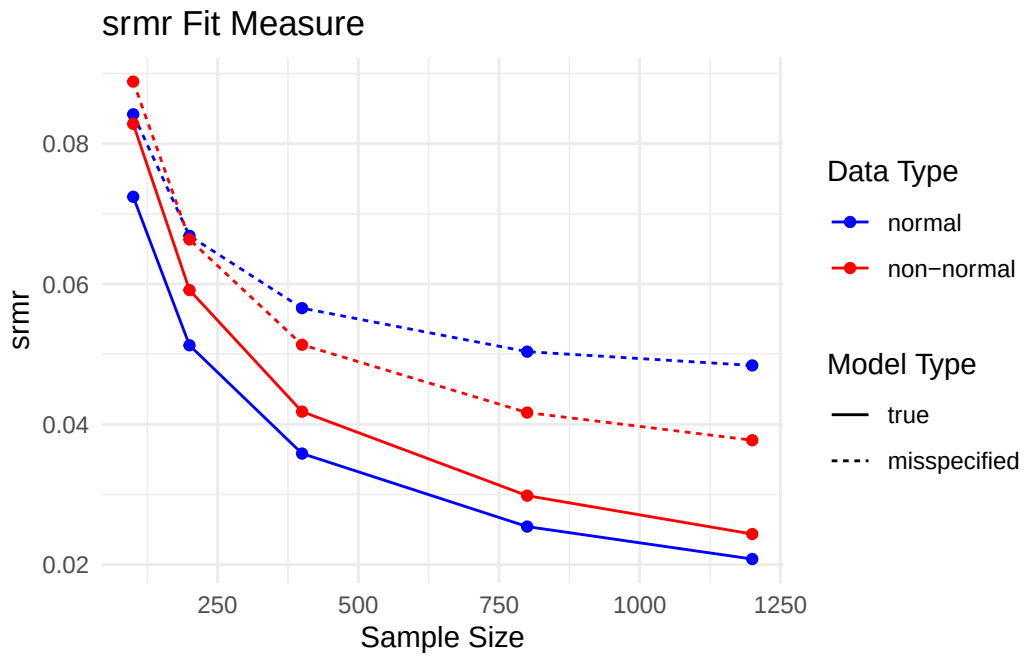
```r
# Plotting it
plot_fit_measures <- function(measure_name) {
    p <- ggplot(res_long %>% filter(measure == measure_name), aes(x = sample_size, y = val
        geom_line() +
        geom_point() +
        labs(title = paste(measure_name, "Fit Measure"),
             x = "Sample Size",
             y = measure_name,
             color = "Data Type",
             linetype = "Model Type") +
        scale_color_manual(values = c("normal" = "blue", "non-normal" = "red")) +
        theme_minimal()

    print(p)
}

# Generate plots for each fit measure
fit_measures <- unique(res_long$measure)
```

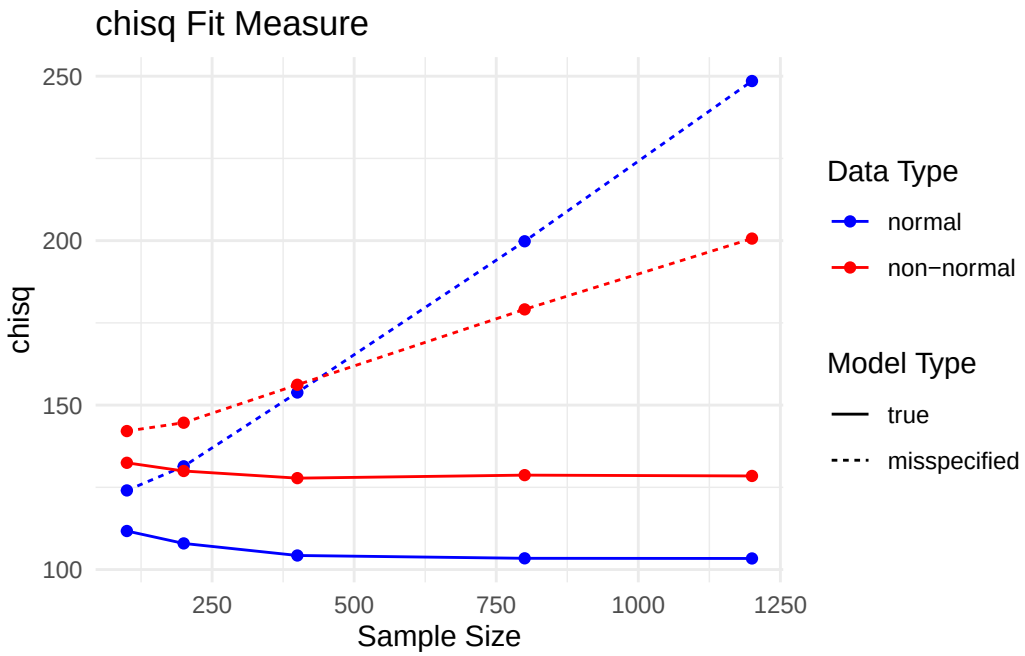
```r
for (measure in fit_measures) {
    plot_fit_measures(measure)
}
```




tli Fit Measure
1.00
0.95
0.90
0.85
0.80
tli
250
500
750
1000
1250
Sample Size
Data Type
normal
non−normal
Model Type
true
misspecified


rmsea Fit Measure
0.06
0.04
0.02
rmsea
250
500
750
1000
1250
Sample Size
Data Type
normal
non−normal
Model Type
true
misspecified


srmr Fit Measure
srmr
0.08
0.06
0.04
0.02
250
500
750
1000
1250
Sample Size
Data Type
normal
non−normal
Model Type
true
misspecified

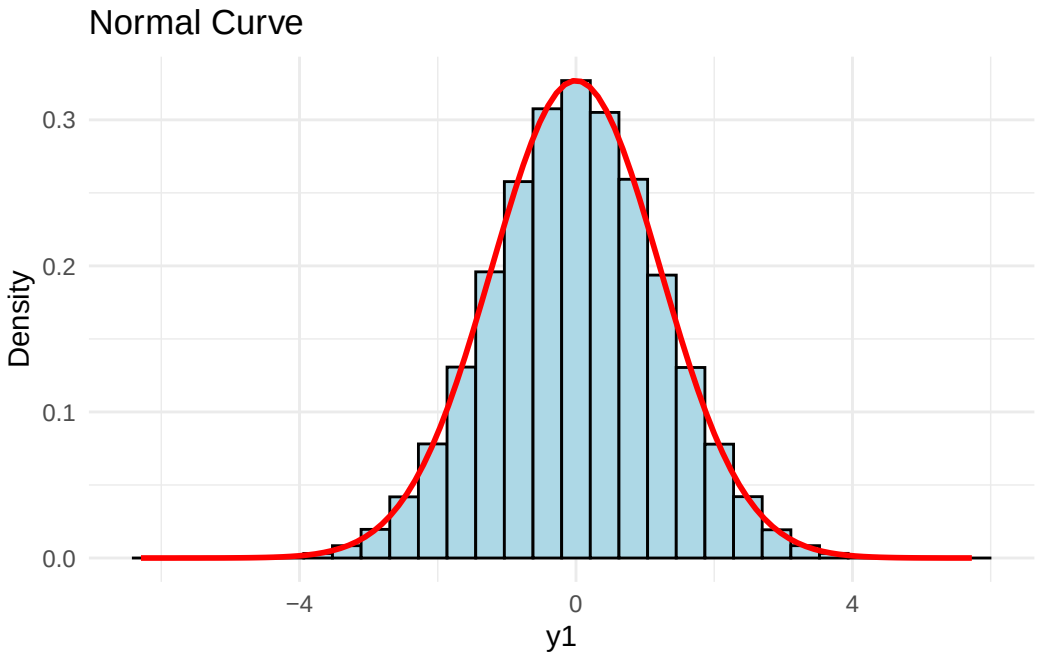

chisq Fit Measure
chisq
250
200
150
100
250
500
750
1000
1250
Sample Size
Data Type
normal
non−normal
Model Type
true
misspecified

Normal!

```
normal_simulated_data <- simulateData(
  true_model, sample.nobs = 1e6, skewness = 0, kurtosis = 0
  )
good_var <- normal_simulated_data$y1

ggplot(normal_simulated_data, aes(x = y1)) +
  geom_histogram(aes(y = ..density..), bins = 30, fill = "lightblue", color = "black") +
  stat_function(fun = dnorm, args = list(mean = mean(good_var), sd = sd(good_var)), color
  labs(title = "Normal Curve", x = "y1", y = "Density") +
  theme_minimal()
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```

```
Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
i Please use `after_stat(density)` instead.
```



Non-normal...

```
non_normal_simulated_data <- simulateData(
  true_model, sample.nobs = 1e6, skewness = 2, kurtosis = 6
  )

bad_var <- non_normal_simulated_data$y1

ggplot(non_normal_simulated_data, aes(x = y1)) +
  geom_histogram(aes(y = ..density..), bins = 30, fill = "lightblue", color = "black") +
  stat_function(fun = dnorm, args = list(mean = mean(bad_var), sd = sd(bad_var)), color =
  labs(title = "Non-Normal Curve", x = "y1", y = "Density") +
  theme_minimal()
```