Solutions to Exercises

Portfolio Optimization: Theory and Application
Chapter 9 – High-Order Portfolios

Daniel P. Palomar (2025). *Portfolio Optimization: Theory and Application.*
Cambridge University Press.

portfoliooptimizationbook.com

Exercise 9.1: Non-Gaussian return distribution

a. Download market data for one asset.
b. Plot the histograms for different frequencies of returns.
c. Try to fit a Gaussian distribution.
d. Assess the asymmetry as well as the thickness of the tails for these histograms (use Q–Q plots, compute skewness and kurtosis, etc.).

Solution

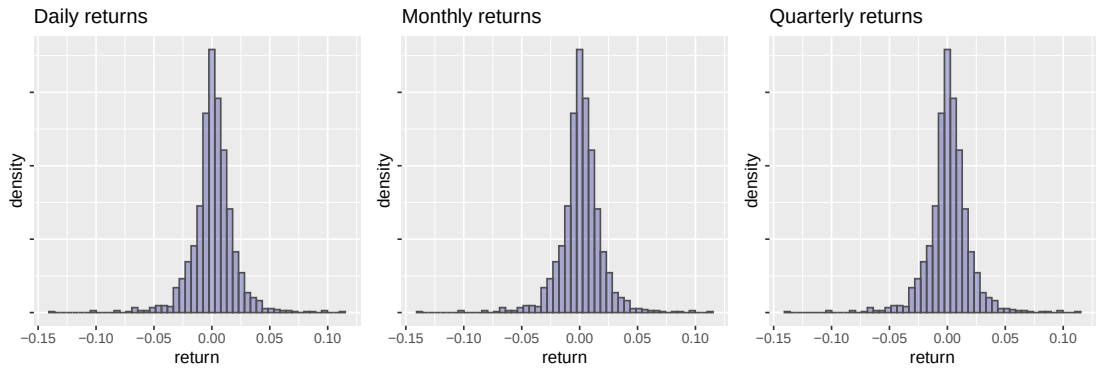
a. Download market data for one asset:

```
library(quantmod)
library(pob)       # Market data used in the book

# # Get data from Yahoo!Finance
# prices  <- Ad(getSymbols("AAPL",
#                          from = "2015-01-01", to = "2020-09-22",
#                          auto.assign = FALSE))

# Use book data
data(SP500_2015to2020)
prices <- SP500_2015to2020$stocks[, "AAPL"]
```

b. Histograms for different frequencies of returns:

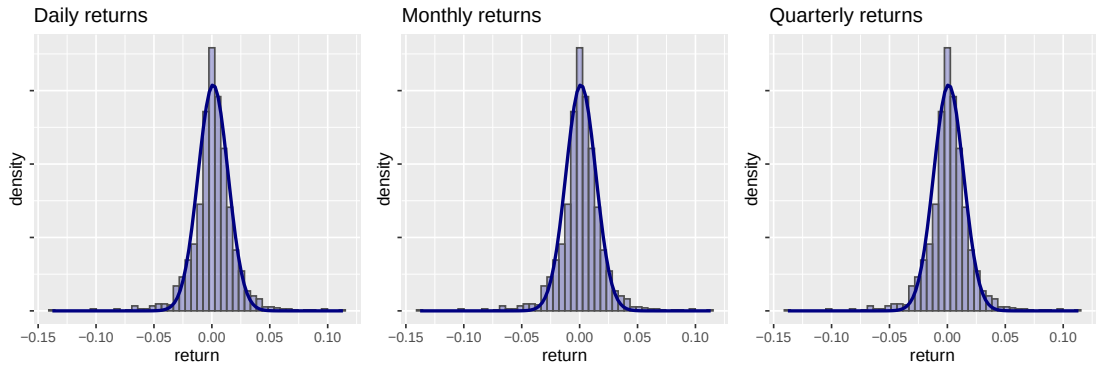
```r
# Compute returns at different frequencies
dailyreturns     <- na.omit(diff(log(prices)))
monthlyreturns   <- na.omit(diff(log(prices), 21))
quarterlyreturns <- na.omit(diff(log(prices), 63))
```

```r
library(ggplot2)
library(patchwork)

plot_histogram <- function(returns) {
  ggplot(fortify(returns, melt = TRUE), aes(x = Value)) +
    geom_histogram(aes(y = after_stat(density)), bins = 50,
                   fill = "darkblue", alpha = 0.3, col = "gray31") +
    theme(axis.text.y = element_blank()) +
    xlab("return") + ylab("density")
}

p1 <- plot_histogram(dailyreturns) + ggtitle("Daily returns")
p2 <- plot_histogram(dailyreturns) + ggtitle("Monthly returns")
p3 <- plot_histogram(dailyreturns) + ggtitle("Quarterly returns")
p1 | p2 | p3
```



c. Histogramps with a Gaussian fit:

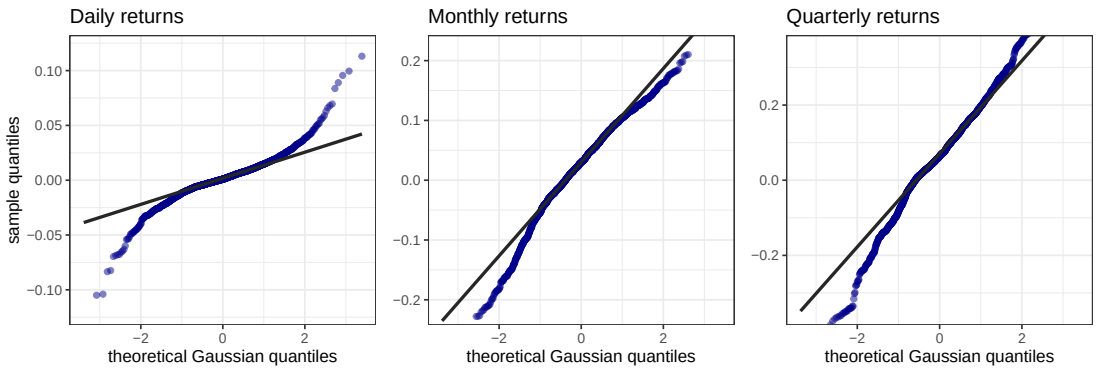
```
plot_histogram_and_Gaussian_fit <- function(returns) {
  ggplot(fortify(returns, melt = TRUE), aes(x = Value)) +
    geom_histogram(aes(y = after_stat(density)), bins = 50,
                   fill = "darkblue", alpha = 0.3, col = "gray31") +
    stat_function(fun = dnorm,
                  args = list(mean = mean(returns, na.rm = TRUE),
                              sd = 0.7*sd(returns, na.rm = TRUE)),
                  color = "navy", linewidth = 1) +
    theme(axis.text.y = element_blank()) +
    xlab("return") + ylab("density")
}

p1 <- plot_histogram_and_Gaussian_fit(dailyreturns) + ggtitle("Daily returns")
p2 <- plot_histogram_and_Gaussian_fit(dailyreturns) + ggtitle("Monthly returns")
p3 <- plot_histogram_and_Gaussian_fit(dailyreturns) + ggtitle("Quarterly returns")
p1 | p2 | p3
```



d. Assess the asymmetry as well as the thickness of the tails for these histograms (use Q–Q plots, compute skewness and kurtosis, etc.).

Q–Q plots of log-returns at different frequencies:

```
p1 <- ggplot(data.frame(y = as.vector(dailyreturns)), aes(sample = y)) +
  stat_qq(color = "darkblue", alpha = 0.5) +
  stat_qq_line(color = "gray15", linewidth = 1.0) +
  theme_bw() +  coord_cartesian(ylim = c(-0.12, 0.12)) +
  labs(title = "Daily returns", x = "theoretical Gaussian quantiles", y = "sample quantiles")
p2 <- ggplot(data.frame(y = as.vector(monthlyreturns)), aes(sample = y)) +
  stat_qq(color = "darkblue", alpha = 0.5) +
  stat_qq_line(color = "gray15", linewidth = 1.0) +
  theme_bw() + coord_cartesian(ylim = c(-0.22, 0.22)) +
  labs(title = "Monthly returns", x = "theoretical Gaussian quantiles", y = NULL)
p3 <- ggplot(data.frame(y = as.vector(quarterlyreturns)), aes(sample = y)) +
  stat_qq(color = "darkblue", alpha = 0.5) +
  stat_qq_line(color = "gray15", linewidth = 1.0) +
  theme_bw() +  coord_cartesian(ylim = c(-0.35, 0.35)) +
  labs(title = "Quarterly returns", x = "theoretical Gaussian quantiles", y = NULL)

p1 | p2 | p3
```

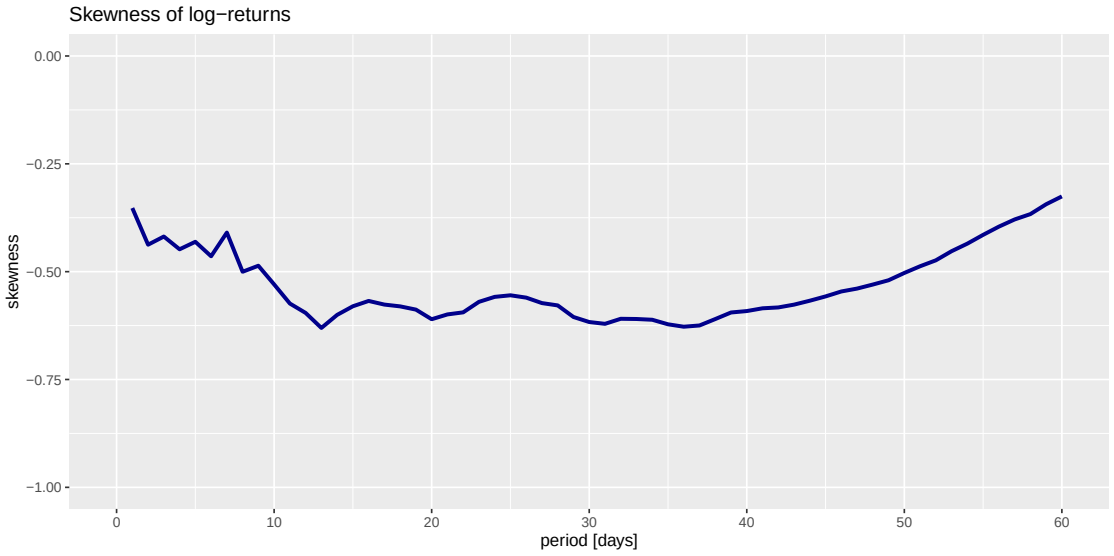


Skewness and kurtosis:

```
library(PerformanceAnalytics)

# compute kurtosis vs frequency (observe aggregational Gaussianity)
asset_kurtosis <- NULL
asset_skewness <- NULL
for(l in 1:60) {
  asset_kurtosis[l] <- kurtosis(diff(log(prices), l), method="excess")
  asset_skewness[l] <- skewness(diff(log(prices), l))
}
```

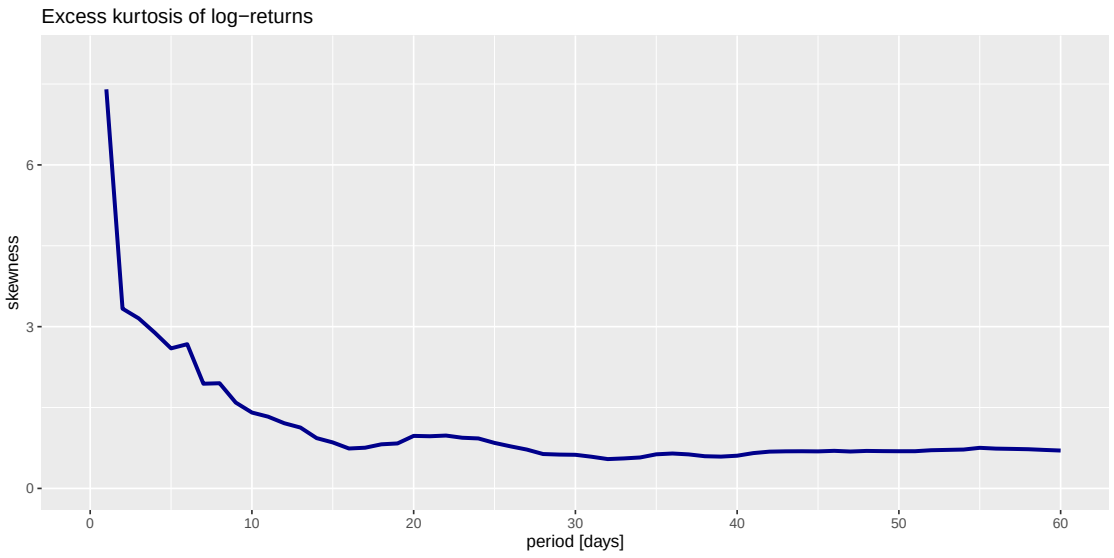
- The skewness is negative, indicating an asymmetry with heavier left tails:

```
ggplot(data.frame("x" = 1:60, "y" = asset_skewness), aes(x, y)) +
  geom_line(color = "darkblue", linewidth = 1.2) +
  scale_x_continuous(limits = c(0, 60), breaks = seq(0, 60, by = 10)) +
  coord_cartesian(ylim = c(-1, 0)) +
  labs(title = "Skewness of log-returns", x = "period [days]", y = "skewness")
```



- There is a clear excess kurtosis, indicating clear heavy tails:

```
ggplot(data.frame("x" = 1:60, "y" = asset_kurtosis), aes(x, y)) +
  geom_line(color = "darkblue", linewidth = 1.2) +
  scale_x_continuous(limits = c(0, 60), breaks = seq(0, 60, by = 10)) +
  scale_y_continuous(limits = c(0, 8), breaks = seq(0, 12, by = 3)) +
  labs(title = "Excess kurtosis of log-returns", x = "period [days]", y = "skewness")
```



Exercise 9.2: Computation of portfolio sample moments

a. Download market data corresponding to N assets during a period with T observations, $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_T \in \mathbb{R}^N$.
b. Estimate the mean vector, covariance matrix, co-skewness matrix, and co-kurtosis matrix of the data via sample means.
c. Design some portfolio, such as the $1/N$ portfolio, and compute the four moments of the portfolio returns (i.e., mean, variance, skewness, and kurtosis).
d. Additionally, compute the gradient and Hessian of the four portfolio moments.
e. Repeat the whole process for different values of N, while keeping track of the computational cost, and make a final plot of complexity vs. N.

Solution

a. Market data corresponding to N stocks:

```
library(xts)
library(pob)       # Market data used in the book

# Use data from package pob
data(SP500_2015to2020)
stock_prices <- SP500_2015to2020$stocks[
  "2019-10::",
  c("AAPL", "AMZN", "AMD", "GM", "GOOGL", "MGM", "MSFT", "QCOM", "TSCO", "UPS")
  ]
X <- diff(log(stock_prices))[-1]
N <- ncol(X)
```

b. Estimate the mean vector, covariance matrix, co-skewness matrix, and co-kurtosis matrix of the data via sample means:

```
library(highOrderPortfolios)

# Via the package highOrderPortfolios
X_moments <- estimate_sample_moments(X)
names(X_moments)
```

```
[1] "mu"        "Sgm"       "Phi_mat"   "Psi_mat"   "Phi"       "Psi"
[7] "Phi_shred" "Psi_shred"
```

```
length(X_moments$mu)
```

```
[1] 10
```

```
dim(X_moments$Sgm)
```

```
[1] 10 10
```

```
dim(X_moments$Phi_mat)
```

```
[1]  10 100
```

```
dim(X_moments$Psi_mat)
```

```
[1]   10 1000
```

```
# Via the package PerformanceAnalytics
library(PerformanceAnalytics)

mu <- colMeans(X)
Sgm <- cov(X)
Phi_mat <- CoSkewnessMatrix(X)
Psi_mat <- CoKurtosisMatrix(X)
```

c. Design some portfolio, such as the $1/N$ portfolio, and compute the four moments of the portfolio returns (i.e., mean, variance, skewness, and kurtosis):

```
library(PerformanceAnalytics)

w <- rep(1/N, N)
rets <- X %*% w

# Via the package highOrderPortfolios
w_moments <- eval_portfolio_moments(w, X_moments)
w_moments
```

```
         mean      variance      skewness      kurtosis
 1.528074e-03  6.056852e-04 -1.790314e-05  4.558928e-06
```

```
# Via the package PerformanceAnalytics
colMeans(rets)
```

```
[1] 0.001528074
```

```
cov(rets)
```

```
              [,1]
[1,] 0.0006056852
```

```
CoSkewnessMatrix(rets)
```

```
               [,1]
[1,] -1.790314e-05
```

```
CoKurtosisMatrix(rets)
```

```
              [,1]
[1,] 4.558928e-06
```

d. Gradient and Hessian of the four portfolio moments:

$$
\begin{aligned}
\nabla \phi_1(\boldsymbol{w}) &= \boldsymbol{\mu}, \\
\nabla \phi_2(\boldsymbol{w}) &= 2\,\boldsymbol{\Sigma}\boldsymbol{w}, \\
\nabla \phi_3(\boldsymbol{w}) &= 3\,\boldsymbol{\Phi}(\boldsymbol{w} \otimes \boldsymbol{w}), \\
\nabla \phi_4(\boldsymbol{w}) &= 4\,\boldsymbol{\Psi}(\boldsymbol{w} \otimes \boldsymbol{w} \otimes \boldsymbol{w})
\end{aligned}
$$

and

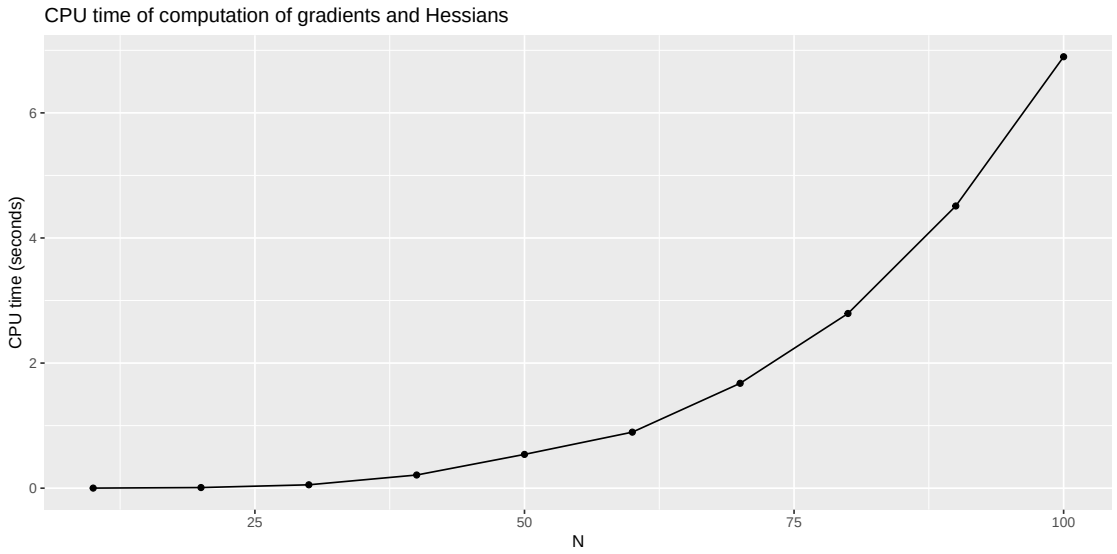
$$
\begin{aligned}
\nabla^2 \phi_1(\boldsymbol{w}) &= \mathbf{0}, \\
\nabla^2 \phi_2(\boldsymbol{w}) &= 2\,\boldsymbol{\Sigma}, \\
\nabla^2 \phi_3(\boldsymbol{w}) &= 6\,\boldsymbol{\Phi}(\boldsymbol{I} \otimes \boldsymbol{w}), \\
\nabla^2 \phi_4(\boldsymbol{w}) &= 12\,\boldsymbol{\Psi}(\boldsymbol{I} \otimes \boldsymbol{w} \otimes \boldsymbol{w}),
\end{aligned}
$$

```r
# Gradients of the portfolio w
grad_phi1 <- mu
grad_phi2 <- 2 * Sgm %*% w
grad_phi3 <- 3 * Phi_mat %*% kronecker(w, w)
grad_phi4 <- 4 * Psi_mat %*% kronecker(w, kronecker(w, w))

# Hessians of the portfolio w
hess_phi1 <- matrix(0, N, N)
hess_phi2 <- 2 * Sgm
hess_phi3 <- 6 * Phi_mat %*% kronecker(diag(N), w)
hess_phi4 <- 12 * Psi_mat %*% kronecker(diag(N), kronecker(w, w))
```

e. Complexity vs. N:

```r
N_sweep <- seq(10, 100, by = 10)
cpu_time <- NULL
for (N in N_sweep) {
  # Get data
  stock_prices <- SP500_2015to2020$stocks[, 1:N]
  X <- diff(log(stock_prices))[-1]

  start_time <- Sys.time()
  # Compute matrix moments
  mu <- colMeans(X)
  Sgm <- cov(X)
  Phi_mat <- CoSkewnessMatrix(X)
  Psi_mat <- CoKurtosisMatrix(X)

  w <- rep(1/N, N)

  # Gradients of the portfolio w
  grad_phi1 <- mu
  grad_phi2 <- 2 * Sgm %*% w
  grad_phi3 <- 3 * Phi_mat %*% kronecker(w, w)
  grad_phi4 <- 4 * Psi_mat %*% kronecker(w, kronecker(w, w))

  # Hessians of the portfolio w
  hess_phi1 <- matrix(0, N, N)
  hess_phi2 <- 2 * Sgm
  hess_phi3 <- 6 * Phi_mat %*% kronecker(diag(N), w)
  hess_phi4 <- 12 * Psi_mat %*% kronecker(diag(N), kronecker(w, w))

  cpu_time <- c(cpu_time, Sys.time() - start_time)
}
```

```
data.frame(
  N_sweep = N_sweep,
  cpu_time = cpu_time
  ) |>
  ggplot(aes(x = N_sweep, y = cpu_time)) +
  geom_point() +geom_line() +
  labs(title = "CPU time of computation of gradients and Hessians", x = "N", y = "CPU time (seconds
```



Exercise 9.3: Comparison of nonparametric, structured, and parametric moments

a. Download market data corresponding to N assets during a period with T observations, $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_T \in \mathbb{R}^N$.
b. Design some portfolio, such as the $1/N$ portfolio.
c. Estimate the mean, variance, skewness, and kurtosis of the portfolio returns in the following ways:
   - nonparametric moments: via a direct sample mean estimation of the mean vector, covariance matrix, co-skewness matrix, and co-kurtosis matrix;
   - structured moments: via fitting a single market-factor model to the returns;
   - parametric moments: via fitting a multivariate skew t distribution to the returns.
d. Repeat the whole process for different values of N, while keeping track of the computational

cost, and make a final plot of complexity vs. N.

Solution

a. Market data corresponding to N stocks:

```
library(xts)
library(pob)       # Market data used in the book

# Use data from package pob
data(SP500_2015to2020)
stock_prices <- SP500_2015to2020$stocks[
  "2019-10::",
  c("AAPL", "AMZN", "AMD", "GM", "GOOGL", "MGM", "MSFT", "QCOM", "TSCO", "UPS")
  ]
X <- diff(log(stock_prices))[-1]
N <- ncol(X)
T <- nrow(X)
```

b. Design some portfolio, such as the $1/N$ portfolio.

```
w <- rep(1/N, N)
rets <- X %*% w
```

c. Estimation of the mean, variance, skewness, and kurtosis of the portfolio returns:

- Nonparametric moments: via a direct sample mean estimation of the mean vector, covariance matrix, co-skewness matrix, and co-kurtosis matrix:

```
# Via the package highOrderPortfolios
X_sample_moments <- estimate_sample_moments(X)
eval_portfolio_moments(w, X_sample_moments)
```

```
         mean      variance      skewness      kurtosis
 1.528074e-03  6.056852e-04 -1.790314e-05  4.558928e-06
```

```
# Or directly
(mean <- as.numeric(t(w) %*% X_sample_moments$mu))
```

```
[1] 0.001528074
```

```
(variance <- as.numeric(t(w) %*% X_sample_moments$Sgm %*% w))
```

```
[1] 0.0006056852
```

```r
(skewness <- as.numeric(t(w) %*% X_sample_moments$Phi_mat %*% kronecker(w, w)))
```

```
[1] -1.790314e-05
```

```r
(kurtosis <- as.numeric(t(w) %*% X_sample_moments$Psi_mat %*% kronecker(w, kronecker(w, w))))
```

```
[1] 4.558928e-06
```

- Structured moments: via fitting a single market-factor model to the returns:

```r
library(covFactorModel)  # devtools::install_github("dppalomar/covFactorModel")

# First, fit the factor model
factor_model <- factorModel(X, K = 1)
names(factor_model)
```

```
[1] "alpha"    "beta"     "factors"  "residual"
```

```r
# sanity check
X_ <- with(factor_model,
           matrix(alpha, T, N, byrow = TRUE) + factors %*% t(beta) + residual)
norm(X - X_, "F")
```

```
[1] 6.350584e-17
```

```r
# Moments of (univariate) factor
phi1_f <- colMeans(factor_model$factors)
phi2_f <- as.numeric(cov(factor_model$factors))
phi3_f <- as.numeric(CoSkewnessMatrix(factor_model$factors))
phi4_f <- as.numeric(CoKurtosisMatrix(factor_model$factors))

# Direct computation of moments
beta <- factor_model$beta
(mean <- factor_model$alpha + beta * phi1_f)
```

```
           factor1
AAPL   0.0028402022
AMZN   0.0023956434
AMD    0.0040401208
GM    -0.0007374723
GOOGL  0.0007764392
MGM   -0.0009570935
MSFT   0.0017299760
QCOM   0.0017869172
TSCO   0.0019044821
UPS    0.0015015205
```

```
(covmat <- beta %*% t(beta) * phi2_f + cov(factor_model$residual))
```

```
              AAPL          AMZN          AMD           GM        GOOGL
AAPL  0.0008822256 0.0004768503 0.0007781226 0.0006087938 0.0006113070
AMZN  0.0004768503 0.0005320405 0.0005634504 0.0002745291 0.0004200361
AMD   0.0007781226 0.0005634504 0.0014462196 0.0005797390 0.0006295242
GM    0.0006087938 0.0002745291 0.0005797390 0.0014963563 0.0005865125
GOOGL 0.0006113070 0.0004200361 0.0006295242 0.0005865125 0.0006366822
MGM   0.0007456080 0.0003320931 0.0007401952 0.0015930452 0.0007395355
MSFT  0.0007470387 0.0004984805 0.0007758925 0.0006208347 0.0006513101
QCOM  0.0007549356 0.0004715872 0.0008395162 0.0007282633 0.0006638048
TSCO  0.0004140694 0.0002862020 0.0004875704 0.0003847843 0.0003768257
UPS   0.0004754393 0.0002848051 0.0004834530 0.0004255210 0.0003943285
               MGM          MSFT          QCOM          TSCO          UPS
AAPL  0.0007456080 0.0007470387 0.0007549356 0.0004140694 0.0004754393
AMZN  0.0003320931 0.0004984805 0.0004715872 0.0002862020 0.0002848051
AMD   0.0007401952 0.0007758925 0.0008395162 0.0004875704 0.0004834530
GM    0.0015930452 0.0006208347 0.0007282633 0.0003847843 0.0004255210
GOOGL 0.0007395355 0.0006513101 0.0006638048 0.0003768257 0.0003943285
MGM   0.0029597674 0.0007598067 0.0009499323 0.0005047804 0.0005528630
MSFT  0.0007598067 0.0008386197 0.0007786540 0.0004277344 0.0004403452
QCOM  0.0009499323 0.0007786540 0.0011521229 0.0004480119 0.0005743945
TSCO  0.0005047804 0.0004277344 0.0004480119 0.0005912636 0.0002906866
UPS   0.0005528630 0.0004403452 0.0005743945 0.0002906866 0.0006504941
```

```
skewness_mat <- beta %*% kronecker(t(beta), t(beta)) * phi3_f +
    CoSkewnessMatrix(factor_model$residual)
kurtosis_mat <- beta %*% kronecker(t(beta), kronecker(t(beta), t(beta))) * phi4_f +
    CoKurtosisMatrix(factor_model$residual)
```

- Parametric moments: via fitting a multivariate skew t distribution to the returns:

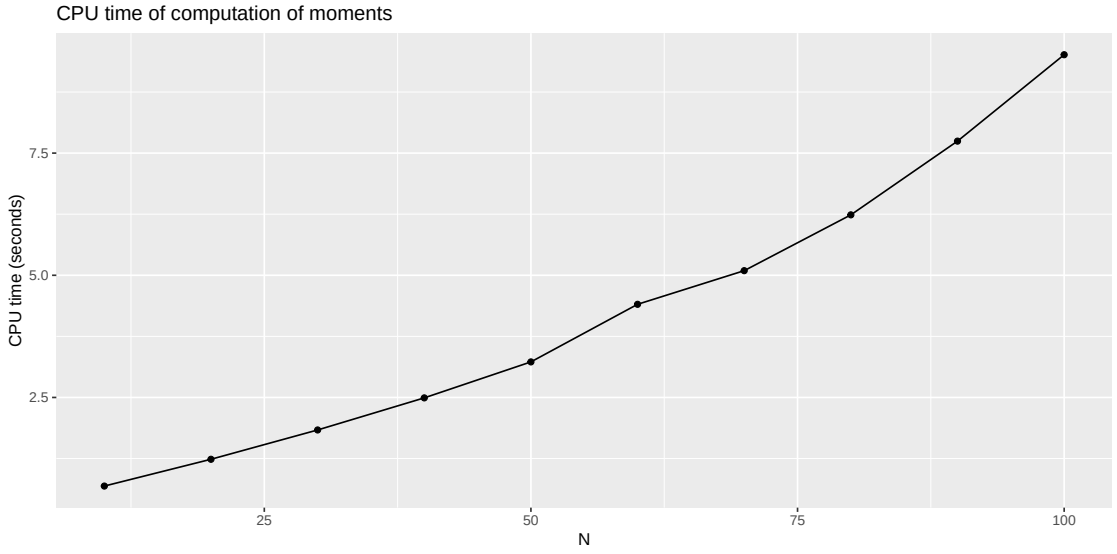
```
# Via the package highOrderPortfolios
X_skew_t_params <- estimate_skew_t(X)
eval_portfolio_moments(w, X_skew_t_params)
```

```
        mean      variance      skewness      kurtosis
 1.574914e-03  3.486179e-04 -9.296309e-08  5.105360e-07
```

d. Complexity vs. N:

```r
N_sweep <- seq(10, 100, by = 10)
cpu_time <- NULL
for (N in N_sweep) {
  # Get data
  stock_prices <- SP500_2015to2020$stocks[, 1:N]
  X <- diff(log(stock_prices))[-1]

  # Portfolio
  w <- rep(1/N, N)

  start_time <- Sys.time()

  # Compute moments Via the package highOrderPortfolios
  X_sample_moments <- estimate_sample_moments(X)
  eval_portfolio_moments(w, X_sample_moments)

  cpu_time <- c(cpu_time, Sys.time() - start_time)
}
```

```r
data.frame(
  N_sweep = N_sweep,
  cpu_time = cpu_time
  ) |>
  ggplot(aes(x = N_sweep, y = cpu_time)) +
  geom_point() +geom_line() +
  labs(title = "CPU time of computation of moments", x = "N", y = "CPU time (seconds)")
```



Exercise 9.4: Sanity check of parametric moment expressions

a. Generate synthetic data according to a multivariate skew t distribution.
b. Design some portfolio, such as the $1/N$ portfolio.
c. Estimate the mean, variance, skewness, and kurtosis of the portfolio returns in the following ways:
   - nonparametric moments: first estimate via sample means the mean vector, covariance matrix, co-skewness matrix, and co-kurtosis matrix of the data, then evaluate the portfolio moments (as well as gradients and Hessians);
   - parametric moments: first fit a multivariate skew t distribution to these synthetic returns, then evaluate the moments with the parametric expressions (as well as gradients and Hessians).
d. Compare the nonparametric and parametric estimations.
e. Repeat the whole process for different numbers of data samples T, and make a final plot of estimators vs. T.

Solution

a. Generate synthetic data according to a multivariate skew t distribution:

```
# First, get realistic mu, Sigma, and nu
N <- 10
stock_prices <- SP500_2015to2020$stocks[, 1:N]
X <- diff(log(stock_prices))[-1]
T <- nrow(X)
mu_true <- rep(0, N)  #colMeans(X)
Sigma_true <- cov(X)
nu <- 4
```

```
library(mvtnorm)

# Generate synthetic data
set.seed(42)
X_many_observations <- rmvt(n = 1e5, delta = mu_true, sigma = (nu-2)/nu * Sigma_true, df = nu)
X <- X_many_observations[1:200, ]
```

b. Design $1/N$ portfolio:

```
# Portfolio
w <- rep(1/N, N)
```

c. Estimate the mean, variance, skewness, and kurtosis of the portfolio returns via nonparametric and parametric methods:

```
# Compute the "true" moments by using "infinite" observations
X_true_moments <- estimate_sample_moments(X_many_observations)
w_true_moments <- eval_portfolio_moments(w, X_true_moments)
print(w_true_moments)
```

```
         mean      variance      skewness      kurtosis
 3.653691e-05  1.943957e-04 -1.681946e-05  2.049438e-05
```

- Nonparametric moments: first estimate via sample means the mean vector, covariance matrix, co-skewness matrix, and co-kurtosis matrix of the data, then evaluate the portfolio moments (as well as gradients and Hessians):

```
# Via the package highOrderPortfolios
X_sample_moments <- estimate_sample_moments(X)
w_sample_moments <- eval_portfolio_moments(w, X_sample_moments)
print(w_sample_moments)
```

```
         mean      variance      skewness      kurtosis
-6.344690e-04  1.698019e-04  1.227939e-07  2.016924e-07
```

- Parametric moments: first fit a multivariate skew t distribution to these synthetic returns, then evaluate the moments with the parametric expressions (as well as gradients and Hessians):

```
# Via the package highOrderPortfolios
X_skew_t_params <- estimate_skew_t(X)
w_skew_t_moments <- eval_portfolio_moments(w, X_skew_t_params)
print(w_skew_t_moments)
```

```
         mean      variance      skewness      kurtosis
-6.048194e-04  1.280711e-04 -3.513405e-08  6.892538e-08
```

d. Compare the nonparametric and parametric estimations:

```
# Comparison of portfolio moments
print(abs(w_sample_moments - w_true_moments))
```

```
        mean     variance     skewness     kurtosis
6.710059e-04 2.459383e-05 1.694225e-05 2.029269e-05
```

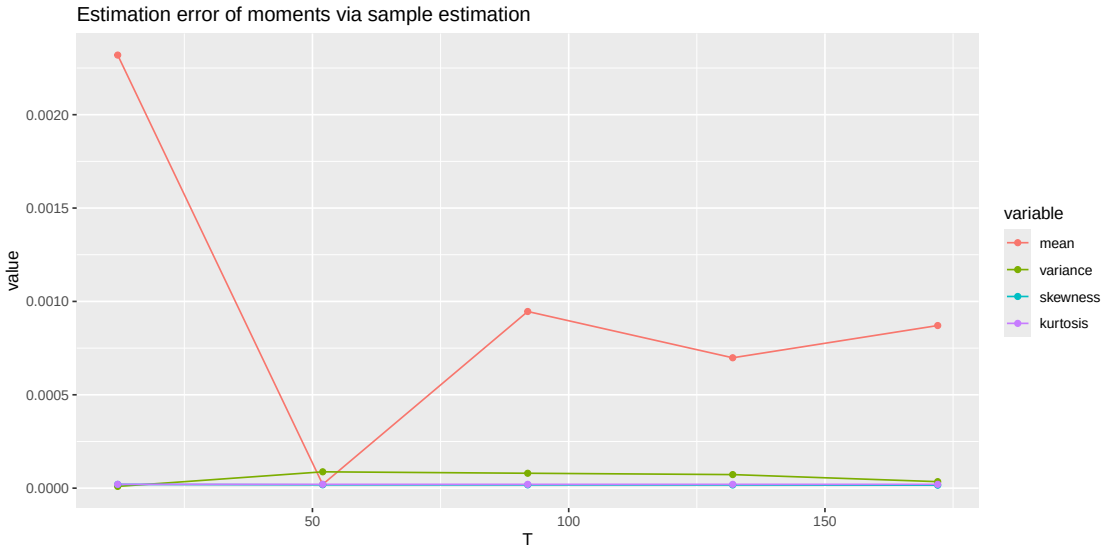
```
print(abs(w_skew_t_moments - w_true_moments))
```

```
        mean     variance     skewness     kurtosis
6.413563e-04 6.632460e-05 1.678433e-05 2.042546e-05
```

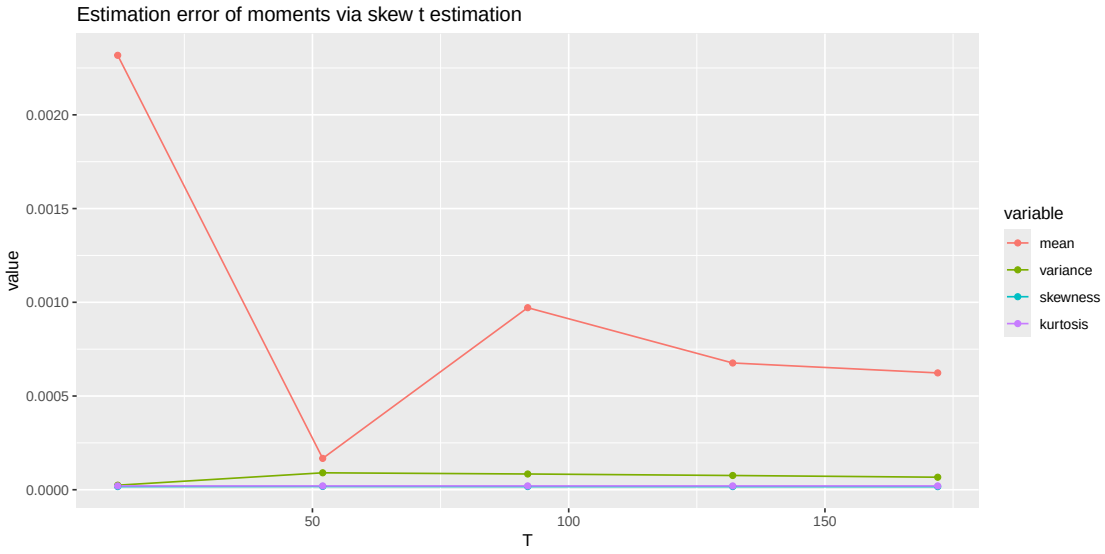
e. Repeat the whole process for different numbers of data samples T:

```r
w_sample_moments_error_df <- data.frame()
w_skew_t_moments_error_df <- data.frame()
T_sweep <- seq(12, 200, by = 40)
for (T in T_sweep) {
  # Get data
  X <- X_many_observations[1:T, ]

  # Compute sample moments
  X_sample_moments <- estimate_sample_moments(X)
  w_sample_moments <- eval_portfolio_moments(w, X_sample_moments)

  # Compute skew t moments
  X_skew_t_params <- estimate_skew_t(X)
  w_skew_t_moments <- eval_portfolio_moments(w, X_skew_t_params)

  # Save results
  new_row <- abs(w_sample_moments - w_true_moments)
  w_sample_moments_error_df <- rbind(w_sample_moments_error_df,
                                     data.frame(T = T, as.list(new_row)))

  new_row <- abs(w_skew_t_moments - w_true_moments)
  w_skew_t_moments_error_df <- rbind(w_skew_t_moments_error_df,
                                     data.frame(T = T, as.list(new_row)))
}
```

```r
library(ggplot2)
library(reshape2)  # for melt function

# Plot for sample moments
w_sample_moments_error_df |>
  melt(id.vars = "T") |>
  ggplot(aes(x = T, y = value, color = variable)) +
  geom_line() + geom_point() +
  labs(title = 'Estimation error of moments via sample estimation')
```




```
# Plot for skew t moments
w_skew_t_moments_error_df |>
  melt(id.vars = "T") |>
  ggplot(aes(x = T, y = value, color = variable)) +
  geom_line() + geom_point() +
  labs(title = 'Estimation error of moments via skew t estimation')
```



Exercise 9.5: L-moments

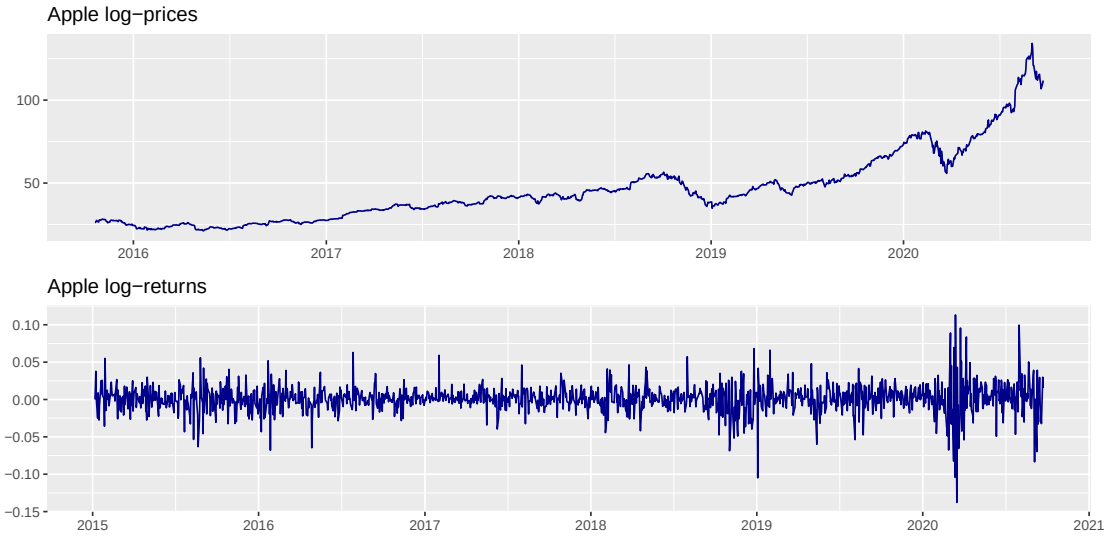
a. Download market data for one asset.
b. Compute the first four moments (i.e., mean, variance, skewness, and kurtosis) in a rolling-window fashion and plot them over time.
c. Compute the first four L-moments (i.e., L-location, L-scale, L-skewness, and L-kurtosis) in a rolling-window fashion and plot them over time.
d. Try different values for the lookback window and compare the regular moments with the L-moments.

Solution

a. Download market data for one asset:

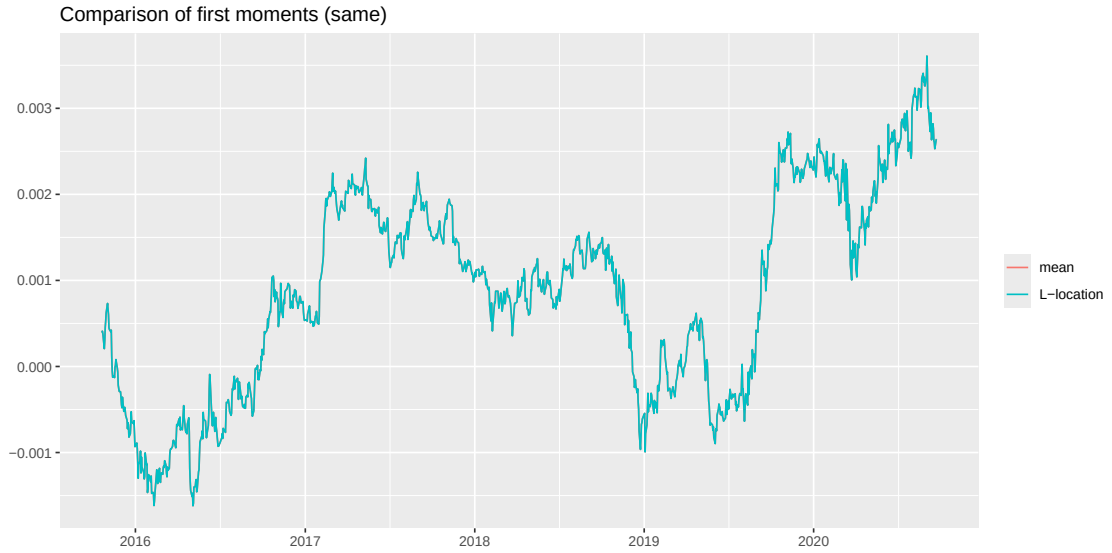
```
library(xts)
library(pob)       # Market data used in the book

# Use book data to get Apple stock
data(SP500_2015to2020)
prices <- SP500_2015to2020$stocks[, "AAPL"]
returns <- diff(log(prices))[-1]
```

```
library(ggplot2)
library(patchwork)

p_prices <- ggplot(fortify(prices["2015-10-20::", ], melt = TRUE),
                   aes(x = Index, y = Value)) +
  geom_line(color = "darkblue", show.legend = FALSE) +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Apple log-prices", x= NULL, y = NULL)

p_returns <- ggplot(fortify(returns, melt = TRUE),
                    aes(x = Index, y = Value)) +
  geom_line(color = "darkblue", show.legend = FALSE) +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Apple log-returns", x= NULL, y = NULL)

p_prices / p_returns
```

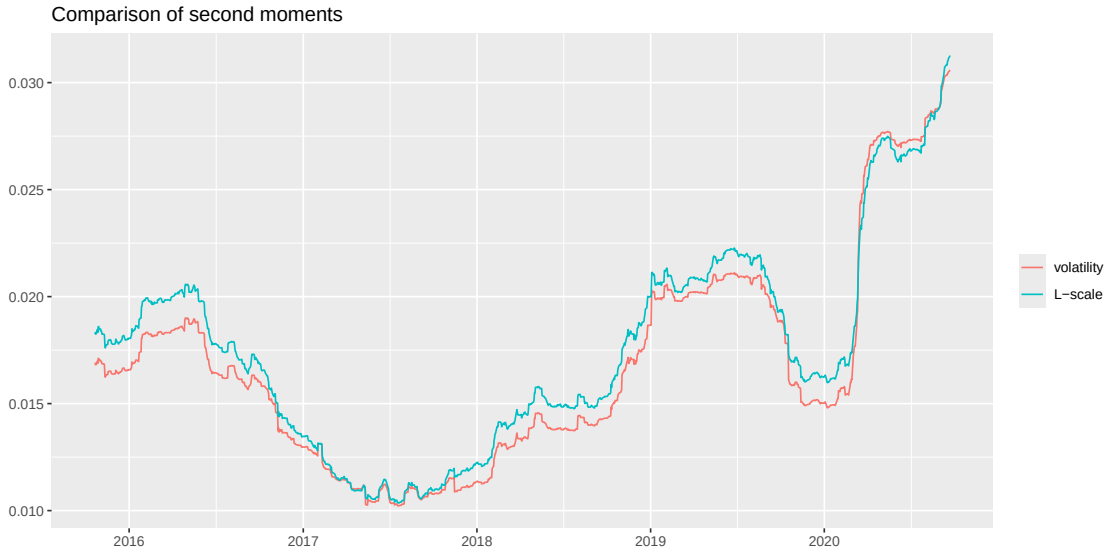


b. and c. First four moments and L-moments over time:

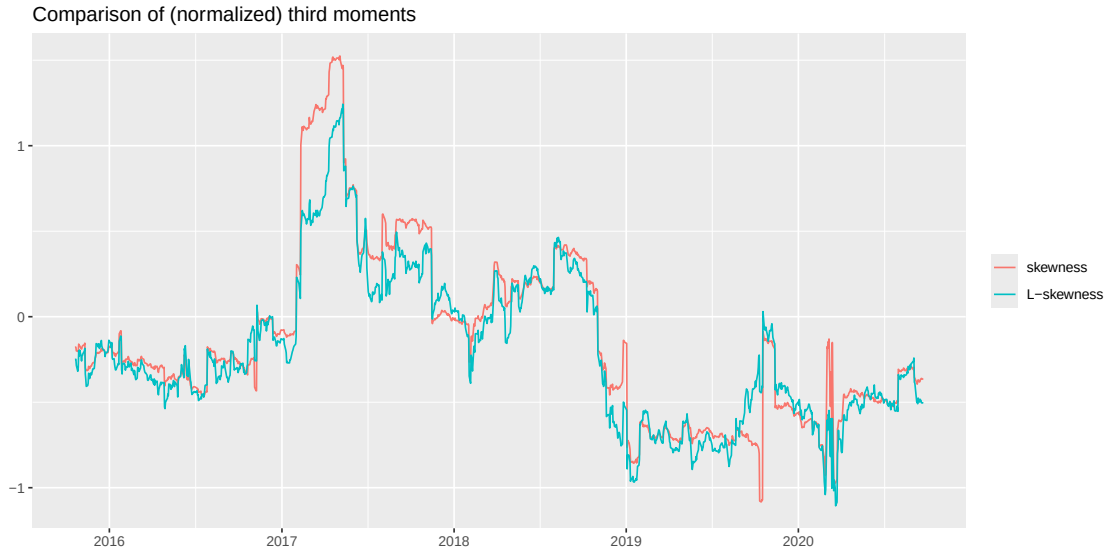
```
library(PerformanceAnalytics)  # for apply.rolling

lookback <- 200

# First moment (same regular moment and L-moment)
mean_rolling <- apply.rolling(returns, width = lookback, by = 1,
                              FUN = function(x) mean(x))

cbind(mean_rolling, mean_rolling)["2015-10-20::", ] |>
  `colnames<-`(c("mean", "L-location")) |>
  fortify(melt = TRUE) |>
  ggplot(aes(x = Index, y = Value, color = Series)) +
  geom_line() +
  labs(title = "Comparison of first moments (same)", x= NULL, y = NULL, color = NULL)
```




```
# Second moments
n <- lookback
i <- 1:n

sd_rolling <- apply.rolling(returns, width = lookback, by = 1, FUN = sd)
weights <- (1/2)*(1/choose(n,2)) * (choose(i-1,1) - choose(n-i,1))
Lmoment2_rolling <- apply.rolling(returns, width = lookback, by = 1,
                                  FUN = function(x) sum(weights*sort(x)))

factor <- 2
cbind(sd_rolling, Lmoment2_rolling * factor)["2015-10-20::", ] |>
  `colnames<-`(c("volatility", "L-scale")) |>
  fortify(melt = TRUE) |>
  ggplot(aes(x = Index, y = Value, color = Series)) +
  geom_line() +
  labs(title = "Comparison of second moments", x= NULL, y = NULL, color = NULL)
```




```
# Third moments
moment3_rolling <- apply.rolling(returns, width = lookback, by = 1,
                                 FUN = function(x) mean((x - mean(x))^3))
weights <- (1/3)*(1/choose(n,3)) * (choose(i-1,2) - 2*choose(i-1,1)*choose(n-i,1) + choose(n-i,2))
Lmoment3_rolling <- apply.rolling(returns, width = lookback, by = 1,
                                  FUN = function(x) sum(weights*sort(x)))

factor <- 10
cbind(moment3_rolling/sd_rolling^3, Lmoment3_rolling/Lmoment2_rolling * factor)["2015-10-20::", ] |
  `colnames<-`(c("skewness", "L-skewness")) |>
  fortify(melt = TRUE) |>
  ggplot(aes(x = Index, y = Value, color = Series)) +
  geom_line() +
  labs(title = "Comparison of (normalized) third moments", x= NULL, y = NULL, color = NULL)
```




```
# Fourth moments
moment4_rolling <- apply.rolling(returns, width = lookback, by = 1,
                                 FUN = function(x) mean((x - mean(x))^4))
weights <- (1/4)*(1/choose(n,4)) * (choose(i-1,3) - 3*choose(i-1,2)*choose(n-i,1) + 3*choose(i-1,1)
Lmoment4_rolling <- apply.rolling(returns, width = lookback, by = 1,
                                  FUN = function(x) sum(weights*sort(x)))

factor <- 20
p_moments4 <- cbind(moment4_rolling/sd_rolling^4, Lmoment4_rolling/Lmoment2_rolling * factor)["2015
  `colnames<-`(c("kurtosis", "L-kurtosis")) |>
  fortify(melt = TRUE) |>
  ggplot(aes(x = Index, y = Value, color = Series)) +
  geom_line() +
  labs(title = "Comparison of (normalized) fourth moments", x= NULL, y = NULL, color = NULL)
```

Exercise 9.6: MVSK portfolios

a. Download market data corresponding to N assets during a period with T observations, $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_T \in \mathbb{R}^N$.
b. Fit a multivariate skew t distribution to the data.
c. Design a traditional mean–variance portfolio.

d. Design a high-order MVSK portfolio.
e. Compare their performance. Try to obtain a clear performance improvement via the introduction of higher orders.

Solution

a. Market data corresponding to N stocks:

```
library(xts)
library(pob)       # Market data used in the book

# Use data from package pob
data(SP500_2015to2020)
stock_prices <- SP500_2015to2020$stocks[
  "2019-10::",
  c("AAPL", "AMZN", "AMD", "GM", "GOOGL", "MGM", "MSFT", "QCOM", "TSCO", "UPS")
  ]
X <- diff(log(stock_prices))[-1]
N <- ncol(X)
```

b. Fit a multivariate skew t distribution to the data:

```
library(highOrderPortfolios)

X_skew_t_params <- estimate_skew_t(X)
names(X_skew_t_params)
```

```
[1] "mu"         "nu"         "gamma"      "scatter"    "chol_Sigma"
[6] "a"
```

c. Design a traditional mean–variance portfolio:

```
# Estimate first- a nd second-order moments
mu <- colMeans(X)
Sigma <- cov(X)
```

```
library(CVXR)

# Define portfolio formulation
design_MVP_scalarized <- function(mu, Sigma, lmd = 1) {
  w <- Variable(nrow(Sigma))
  prob <- Problem(Maximize(t(mu) %*% w - (lmd/2)*quad_form(w, Sigma)),
                  constraints = list(w >= 0, sum(w) == 1))
  result <- solve(prob)
  w <- as.vector(result$getValue(w))
  names(w) <- names(mu)
  return(w)
}

w_MVP <- design_MVP_scalarized(mu, Sigma, lmd = 4)
print(round(w_MVP, digits = 3))
```

```
 AAPL  AMZN   AMD    GM GOOGL   MGM  MSFT  QCOM  TSCO   UPS
0.187 0.398 0.367 0.000 0.000 0.000 0.000 0.000 0.048 0.000
```

d. Design a high-order MVSK portfolio:

```
library(highOrderPortfolios)

# moment weights
gamma <- 4
lambda <- c(1, gamma/2, gamma*(gamma + 1)/6, gamma*(gamma + 1)*(gamma + 2)/24)
lambda <- c(1, 4/2, 10, 20)

# Via sample moments
X_moments <- estimate_sample_moments(X)
sol_moments <- design_MVSK_portfolio_via_sample_moments(lambda, X_moments)
print(round(sol_moments$w, digits = 3))
```

```
 [1] 0.117 0.524 0.316 0.000 0.000 0.000 0.000 0.000 0.043 0.000
```

```
# sanity check with only first two moments (compare with mean-variance)
sol_12moments <- design_MVSK_portfolio_via_sample_moments(c(1, 4/2, 0, 0), X_moments)
print(round(sol_12moments$w, digits = 3))
```

```
 [1] 0.187 0.398 0.367 0.000 0.000 0.000 0.000 0.000 0.048 0.000
```

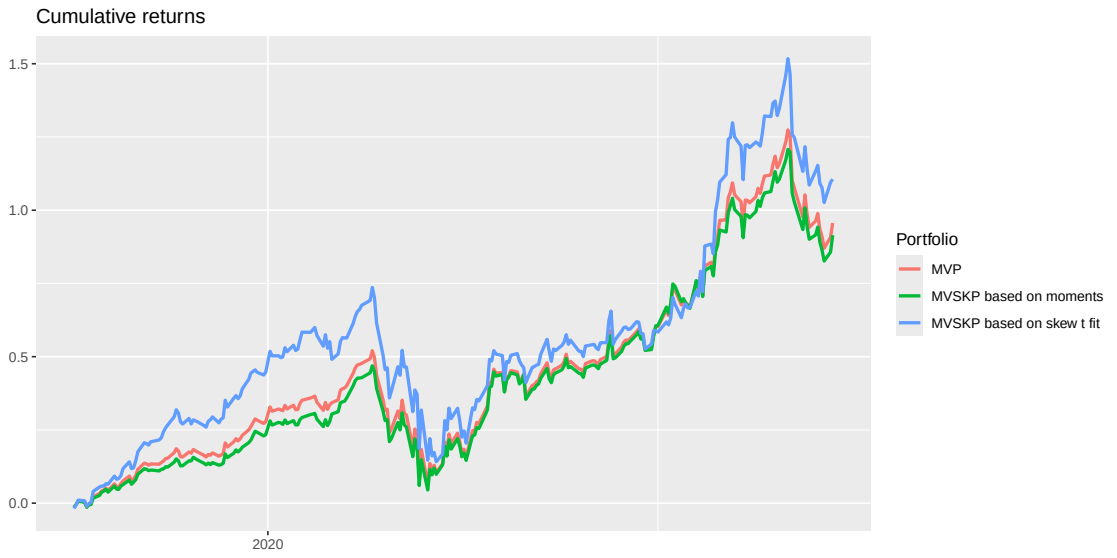
```
# Via skew t fitting
sol_skew_t <- design_MVSK_portfolio_via_skew_t(lambda, X_skew_t_params, method = "SQUAREM")
print(round(sol_skew_t$w, digits = 3))
```

```
[1] 0.361 0.000 0.595 0.000 0.000 0.000 0.000 0.000 0.044 0.000
```

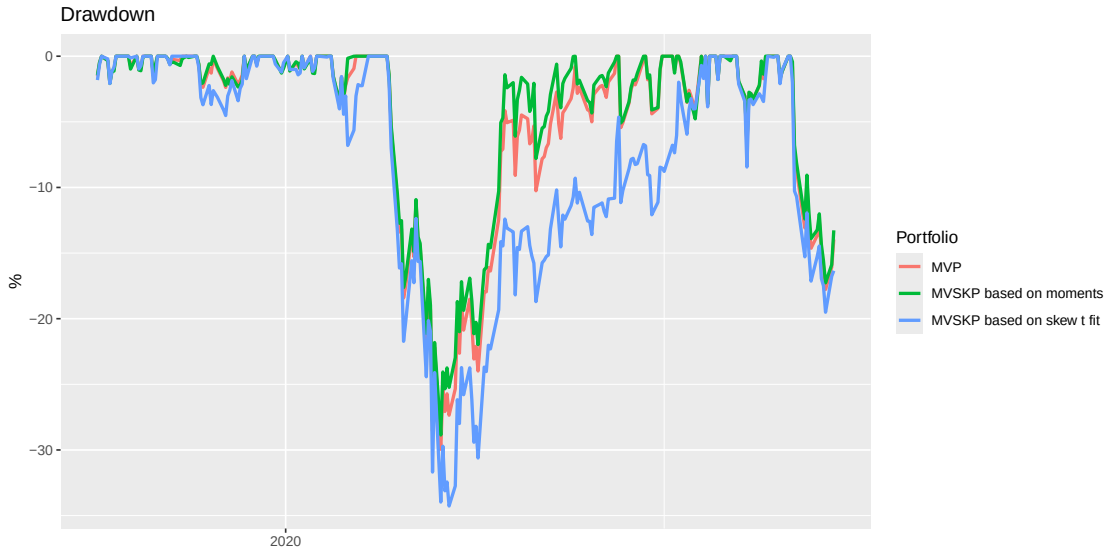
e. Performance comparison of mean–variance portfolio (MVP) and MVSK portfolios based on sample moments or skew t fitting (the performance highly depends on the choice of the hyper-parameters λ_1, λ_2, λ_3, and λ_4):

```
# Naive in-sample backtest assuming daily rebalancing and no transaction cost (:-O)
rets_MVP <- xts(X %*% w_MVP, index(X))
rets_MVSKP_moments <- xts(X %*% sol_moments$w, index(X))
rets_MVSKP_skew_t <- xts(X %*% sol_skew_t$w, index(X))

rets_all <- cbind(rets_MVP, rets_MVSKP_moments, rets_MVSKP_skew_t)
colnames(rets_all) <- c("MVP", "MVSKP based on moments", "MVSKP based on skew t fit")
```

```
library(ggplot2)

# Plot cumulative returns
cumrets_all <- cumprod(1 + rets_all) - 1

cumrets_all |>
  fortify(melt = TRUE) |>
  ggplot(aes(x = Index, y = Value, color = Series)) +
  geom_line(linewidth = 1) +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Cumulative returns", x = NULL, y = NULL, color = "Portfolio")
```



```r
library(PerformanceAnalytics)

# Plot drawdown
drawdown_all <- 100 * Drawdowns(rets_all)

drawdown_all |>
  fortify(melt = TRUE) |>
  ggplot(aes(x = Index, y = Value, color = Series)) +
  geom_line(linewidth = 1) +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Drawdown", x = NULL, y = "%", color = "Portfolio")
```



Exercise 9.7: Portfolio tilting

a. Download market data corresponding to N assets during a period with T observations, $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_T \in \mathbb{R}^N$.
b. Fit a multivariate skew t distribution to the data.
c. Design some portfolio as a reference.
d. Use the portfolio tilting formulation to improve the reference portfolio.
e. Compare their performance. Try to obtain a clear performance improvement via tilting.

Solution

a. Market data corresponding to N stocks:

```
library(xts)
library(pob)       # Market data used in the book

# Use data from package pob
data(SP500_2015to2020)
stock_prices <- SP500_2015to2020$stocks[
  "2019-10::",
  c("AAPL", "AMZN", "AMD", "GM", "GOOGL", "MGM", "MSFT", "QCOM", "TSCO", "UPS")
  ]
X <- diff(log(stock_prices))[-1]
N <- ncol(X)
```

b. Fit a multivariate skew t distribution to the data:

```
library(highOrderPortfolios)

X_skew_t_params <- estimate_skew_t(X)
names(X_skew_t_params)
```

```
[1] "mu"         "nu"         "gamma"      "scatter"    "chol_Sigma"
[6] "a"
```

c. Use $1/N$ portfolio as a reference:

```
# Reference portfolio
w0 <- rep(1/N, N)
print(round(w0, digits = 3))
```

```
 [1] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
```

d. Use the portfolio tilting formulation to improve the reference portfolio.

```
# estimate moments
X_moments <- estimate_sample_moments(X)
w0_moments <- eval_portfolio_moments(w0, X_moments)
print(w0_moments)
```

```
         mean      variance      skewness      kurtosis
 1.528074e-03  6.056852e-04 -1.790314e-05  4.558928e-06
```

```
sol_tilted <- design_MVSKtilting_portfolio_via_sample_moments(
  d = c(1, 4/2, 10, 20), X_moments, w_init = w0, w0 = w0,
  w0_moments = w0_moments, kappa = 0.3 * sqrt(w0 %*% X_moments$Sgm %*% w0)
  )
print(round(sol_tilted$w, digits = 3))
```

```
 [1] 0.026 0.301 0.035 0.135 0.149 0.001 0.001 0.048 0.090 0.213
```

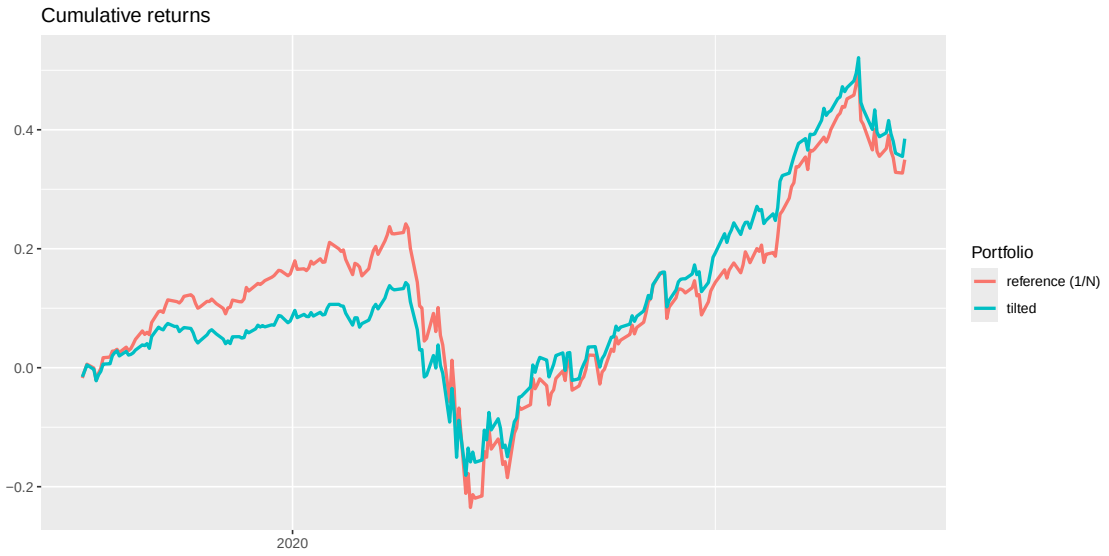
```
print(sol_tilted$moments)
```

```
[1]  1.531614e-03  4.109224e-04 -6.173167e-06  1.435024e-06
```

e. Compare their performance (there is a clear improvement after tilting the reference portfolio):

```
# Naive in-sample backtest assuming daily rebalancing and no transaction cost (:-O)
rets_ref <- xts(X %*% w0, index(X))
rets_tilted <- xts(X %*% sol_tilted$w, index(X))

rets_all <- cbind(rets_ref, rets_tilted)
colnames(rets_all) <- c("reference (1/N)", "tilted")
```

```
library(ggplot2)

# Plot cumulative returns
cumrets_all <- cumprod(1 + rets_all) - 1

cumrets_all |>
  fortify(melt = TRUE) |>
  ggplot(aes(x = Index, y = Value, color = Series)) +
  geom_line(linewidth = 1) +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Cumulative returns", x = NULL, y = NULL, color = "Portfolio")
```



```r
library(PerformanceAnalytics)

# Plot drawdown
drawdown_all <- 100 * Drawdowns(rets_all)

drawdown_all |>
  fortify(melt = TRUE) |>
  ggplot(aes(x = Index, y = Value, color = Series)) +
  geom_line(linewidth = 1) +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  labs(title = "Drawdown", x = NULL, y = "%", color = "Portfolio")
```