STATA FINANCIAL STATISTICS REFERENCE MANUAL
RELEASE 19

A Stata Press Publication
StataCorp LLC
College Station, Texas

Version 19

Published by Stata Press, 4905 Lakeway Drive, College Station, Texas 77845

ISBN-10: 1-59718-461-6
ISBN-13: 978-1-59718-461-8

The suggested citation for this software is

> StataCorp. 2025. *Stata 19*. Statistical software. StataCorp LLC.

The suggested citation for this manual is

> StataCorp. 2025. *Stata 19 Financial Statistics Reference Manual*. College Station, TX: Stata Press.

Contents

[+] This manual includes features that are part of StataNow.

Cross-referencing the documentation

When reading this manual, you will find references to other Stata manuals, for example, **[U] 27 Overview of Stata estimation commands**; [R] **regress**; and [D] **reshape**. The first example is a reference to chapter 27, *Overview of Stata estimation commands*, in the *User's Guide*; the second is a reference to the `regress` entry in the *Base Reference Manual*; and the third is a reference to the `reshape` entry in the *Data Management Reference Manual*.

All the manuals in the Stata Documentation have a shorthand notation:

[GSM]	*Getting Started with Stata for Mac*
[GSU]	*Getting Started with Stata for Unix*
[GSW]	*Getting Started with Stata for Windows*
[U]	*Stata User's Guide*
[R]	*Stata Base Reference Manual*
[ADAPT]	*Stata Adaptive Designs: Group Sequential Trials Reference Manual*
[BAYES]	*Stata Bayesian Analysis Reference Manual*
[BMA]	*Stata Bayesian Model Averaging Reference Manual*
[CAUSAL]	*Stata Causal Inference and Treatment-Effects Estimation Reference Manual*
[CM]	*Stata Choice Models Reference Manual*
[D]	*Stata Data Management Reference Manual*
[DSGE]	*Stata Dynamic Stochastic General Equilibrium Models Reference Manual*
[ERM]	*Stata Extended Regression Models Reference Manual*
[FIN]	*Stata Financial Statistics Reference Manual*
[FMM]	*Stata Finite Mixture Models Reference Manual*
[FN]	*Stata Functions Reference Manual*
[G]	*Stata Graphics Reference Manual*
[H2OML]	*Machine Learning in Stata Using H2O: Ensemble Decision Trees Reference Manual*
[IRT]	*Stata Item Response Theory Reference Manual*
[LASSO]	*Stata Lasso Reference Manual*
[XT]	*Stata Longitudinal-Data/Panel-Data Reference Manual*
[META]	*Stata Meta-Analysis Reference Manual*
[ME]	*Stata Multilevel Mixed-Effects Reference Manual*
[MI]	*Stata Multiple-Imputation Reference Manual*
[MV]	*Stata Multivariate Statistics Reference Manual*
[PSS]	*Stata Power, Precision, and Sample-Size Reference Manual*
[P]	*Stata Programming Reference Manual*
[RPT]	*Stata Reporting Reference Manual*
[SP]	*Stata Spatial Autoregressive Models Reference Manual*
[SEM]	*Stata Structural Equation Modeling Reference Manual*
[SVY]	*Stata Survey Data Reference Manual*
[ST]	*Stata Survival Analysis Reference Manual*
[TABLES]	*Stata Customizable Tables and Collected Results Reference Manual*
[TS]	*Stata Time-Series Reference Manual*
[I]	*Stata Index*
[M]	*Mata Reference Manual*

Intro — Introduction to financial statistics[+]

[+]These features are part of StataNow.

Description Remarks and examples References Also see

Description

Financial statistics provides methods for statistical analysis of financial data. Financial data typically include time series on the returns of financial assets, such as stock price returns, foreign exchange returns, and bond yields. Statistical analysis of financial data often consists of fitting regression models to multiple time series, to cross-sectional data, or to panel data.

This introduction provides an overview of aspects of financial statistics. See [FIN] **fin** for Stata commands and a workflow for the analysis of financial data.

Remarks and examples

Remarks are presented under the following headings:

Introduction

Financial statistics is the branch of statistics concerned with financial data. Financial statistics blends financial theory and statistical methods. Financial theory provides guidance on which financial variables are important to investigate and which relationships among variables are likely to be meaningful and generates empirical implications of theory that can be tested with data. Statistical methods provide the empirical framework for testing theoretical models and provide quantitative measures of qualitative theoretical predictions.

Financial markets are characterized by an abundance of data. We are often interested in asset prices and returns, including equity returns, foreign exchange returns, and bond yields. Financial models propose relationships among these returns, both within and across time, which can be tested. Questions of interest involve both the cross-sectional and the time-series behavior of asset returns. Over any given time frame, some assets have higher returns than others; cross-sectional models attempt to capture this variation. Similarly, returns vary over time, and time-series methods are used to capture this behavior. A perennial question in asset pricing is whether returns can be forecasted using past data, a question that naturally fits into a time-series framework.

Thus, financial statistics blends theory and data, incorporating statistical methods already in use while also designing new techniques to address the particularities of financial data.

For an introduction to financial statistics, see Hurn et al. (2020). Campbell, Lo, and MacKinlay (1997) provide an advanced introduction to the econometrics of financial markets. Cochrane (2005) and Campbell (2018) provide introductions to asset pricing and combine theoretical models and empirical estimation. Boffelli and Urga (2016) provide an introduction to time-series methods in Stata with special attention given to financial data.

Characteristics of financial data

The most basic form of financial data is the price of a financial asset. Asset prices can be recorded at various frequencies, including annually, quarterly, monthly, weekly, daily, and high-frequency intraday. Many assets are traded continuously throughout trading days, so lower-frequency information is an aggregate of higher-frequency observations. This can introduce subtleties in the recording of financial data. For example, a monthly series on an asset price could record the beginning of month value, the middle of month value, the end of month value, or the average of daily observations. In turn, these daily observations could be beginning of day, end of day, or midpoint of daily range. Each of these aggregation methods incorporates different levels of information about the behavior of the asset price and combines that information in different ways. For the most part, in this manual, we will work with daily or monthly asset prices at their end-of-period values.

The return on an asset price is a measure of how that price changes over time. Several types of returns can be defined, and each has its uses. The most basic return is the change in the price, for example, the end-of-day value minus the previous day's end-of-day value. The percentage return is the percentage change in the price, which normalizes for the level of the price and makes comparison over time or across assets more straightforward. Percentage returns are of direct interest to financial market participants but have some properties that make statistical procedures challenging to apply directly. The logarithmic return is the difference in the log asset price, which has similar properties to the percentage return but has the advantage of having properties more suitable for statistical analysis.

The basic form of the data, then, is a collection of observations on one or more asset prices or asset price returns over time. An individual asset price measured over time is a "time series". A collection of asset prices measured over time is a "multiple time series".

Financial statistics is typically concerned with both the cross-sectional and the time-series behavior of asset returns. The cross-section of returns captures the notion that different assets pay different returns at a single point in time, and the researcher would like to understand drivers of this variation in returns. Over time, a researcher would like to know whether returns are forecastable from past returns or other characteristics. Combining the two ideas, a researcher might wish to know whether the variation in returns is persistent over time and which characteristics drive the cross-section of average returns.

Portfolio construction

An individual investing their wealth into a collection of assets is said to be building a "portfolio" of assets. Portfolios are linear combinations of assets with weights indicating shares in a particular asset. If there are, for example, K assets, then the portfolio construction problem is to choose weights $(w_1, \ldots, w_K)$ that satisfy some criterion. An individual might choose to hold a portfolio that achieves a goal that a single asset cannot achieve, such as reducing investment risk. When combined, two assets that tend to move against each other can generate a less variable stream of returns than either asset would in isolation.

The simplest criterion might be to maximize the return of the portfolio. On its own, this criterion would reduce to placing all one's investments into the single best-performing asset. However, if one is willing to sacrifice some average return, then portfolios can typically be constructed that imply lower variance in returns than any single asset. Formally, a minimum variance portfolio is the linear combination of assets that attains the lowest variance possible while still generating a desired target return. The necessary ingredients are the historical average returns of each asset and the covariance matrix of the returns. Collect the average historical returns of the K assets into an $K \times 1$ vector $\mathbf{r} = (r_1, r_2, \ldots, r_K)'$, and collect the historical covariance among the assets into an $K \times K$ covariance matrix $\mathbf{\Sigma}$. Then, for a portfolio with weights $\mathbf{w}$, the mean return and variance of the portfolio are

$$r_p = \mathbf{w}'\mathbf{r}$$
$$\text{Var}(r_p) = \mathbf{w}'\mathbf{\Sigma}\mathbf{w}$$

Suppose the investor wishes to hit a target return $\bar{r}_p$ with a portfolio that has as low a variance as possible. Then the variance-minimizing weights $\widehat{\mathbf{w}}$ that obtain a target return $\bar{r}_p$ are the weights that solve

$$\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \text{Var}(r_p) = \mathbf{w}'\Sigma b\mathbf{w}$$
$$\text{s.t.} \quad \mathbf{w}'\mathbf{r} = \bar{r}_p$$
$$\sum_{i=1}^{K} w_i = 1$$

The first constraint states that the desired return must be $\bar{r}_p$, and the second constraint states that the weights must sum to one.

Another approach maximizes the return-to-risk ratio, also known as the Sharpe ratio. Given a risk-free asset r_f and the standard deviation σ_p of the portfolio, the risk-adjusted return on a portfolio is its Sharpe ratio,

$$S_p = \frac{r_p - r_f}{\sigma_p}$$

The numerator here is the so-called excess return: the return generated by the portfolio in excess of return that can be obtained with negligible risk, say, through investment in secured government debt. The maximum Sharpe ratio strategy chooses weights $\mathbf{w}$ to maximize the Sharpe ratio.

In both the variance minimization and Sharp ratio maximization cases, the weights must sum to one, but there is no requirement for the weights to be positive. A negative weight on an asset indicates that the asset is being "sold short". A short position is the act of borrowing an asset, selling it, and using the proceeds to finance long (positive weight) positions. Then the short position must eventually be closed, at which time the borrowed asset is paid back. Restricting the weights to be positive is thus a restriction on short selling. The above optimization problems can be modified to handle no-short-sales restrictions.

In addition to these optimizing strategies, a researcher might wish to set custom weights or, in the simplest case, to set equal weights on all assets in the portfolio. These settings may be used to assess alternative strategies or provide a baseline against which optimizing strategies can be assessed.

Stata can compute portfolios based on minimum variance or maximum Sharpe ratio, for a given target return or globally, with or without short sales. Stata also provides tools to create custom (nonoptimizing) portfolios and portfolios with equal weights. See [FIN] **finportfolio**.

Portfolio summaries

Individual assets and portfolios can be compared across a number of dimensions. Most of these comparisons involve the mean of returns, their "volatility" (that is, variance), and functions thereof.

Portfolio mean. The portfolio mean is the average return on the portfolio, given the weights and the average historical return on the assets in the portfolio. The units of the mean are the same as the units of the data: with monthly data, we have an average monthly return; with daily data, the average daily return; and so on.

Excess return. The excess return of a portfolio is its return minus the return on a safe asset. Safe assets are those that provide guaranteed or near-guaranteed returns, serving as a benchmark for riskier portfolios. Some securities, like certain short-term bonds, can serve as a risk-free asset. Excess returns represent returns in excess of the risk-free benchmark.

Portfolio standard deviation. The portfolio standard deviation (square root of the variance) captures the variability of the portfolio around its mean and is the simplest measure of portfolio risk.

Market beta. A portfolio's market beta is the regression coefficient in a regression of portfolio returns on market returns. It captures volatility, not in itself, but with respect to the market as a whole. A beta of 1 indicates that when the market is up 1%, the asset or portfolio is also up 1%. A beta of 0 indicates that the portfolio does not move with the market. A negative beta indicates that the portfolio is a "hedge": it tends to rise when the market falls. Betas can be constructed for other characteristics as well, by including them as independent variables in the financial regression.

A portfolio's standard deviation and market beta capture different kinds of risk. Comparing portfolios by their standard deviations penalizes portfolios that are more volatile, regardless of the timing of that volatility. On the other hand, comparing portfolios by their market beta provides more information about risk. Portfolios with high beta tend to produce low returns at the same time as the market as a whole. They also tend to produce high returns when the market overall is also producing high returns. Portfolios with high beta are said to be risky because they fail to pay off when other assets also fail to pay off. By the same token, they can be enticing for those who have an appetite for risk. Assets or portfolios with zero or negative market beta pay off precisely when the market does not, which may be valuable for investors seeking more safety in their portfolio.

Sharpe ratio. The Sharpe ratio, mentioned earlier, divides the excess return of a portfolio by its standard deviation, thus providing a measure of return to risk. A Sharpe ratio of 1, for example, indicates a tradeoff of 1% higher volatility for each 1% in excess returns. A Sharpe ratio of 0.3 indicates a 0.3% excess return for every 1% of higher volatility. A Sharpe ratio of 3 indicates 3% higher excess return for every 1% of higher volatility.

Treynor ratio. The Treynor ratio divides the excess return by the market beta of the portfolio.

Jensen's alpha. Jensen's alpha is the intercept in a regression of the portfolio return on the market return. It captures average return of the portfolio after accounting for the portfolio's covariance with the market.

Additional information about financial regressions, whose outputs are ingredients in market beta, the Treynor ratio, and Jensen's alpha, is presented below. Stata can compute many of these summary statistics automatically after creating a portfolio; see [FIN] **finsummarize**.

Financial regressions

Risk and return

A basic principle in asset pricing is that assets with higher risk ought to compensate with higher returns or, put another way, that any asset exhibiting unusually high returns only does so because it takes on higher risk.

The classic measure of risk is the variance of an asset's return (or its standard deviation). In one direction, assets that have higher returns should have higher risk, or there would be no incentive to hold lower-return assets. In the other direction, assets that are riskier should have higher returns to entice investors to hold such risky assets. Asset-pricing models characterize risk in terms of observable variables. These variables, often called factors or risk factors, are then used as independent variables in models of asset returns.

Work by Sharpe (1964), Lintner (1965), Jensen (1968), and Black (1972) used the mean-variance optimization theory described in *Portfolio construction* to derive the capital asset pricing model (CAPM), in which the appropriate measure of risk is not the variance of an asset's return itself but the covariance of the asset's return with "the market", a measure of the return to all assets. Intuitively, assets that pay off when the market is doing poorly serve as a hedge and can thus trade with lower expected average return; but assets that pay off when the market is doing well require a higher expected return to entice investors to hold them. Financial regressions characterize the exposure of assets to market risk.

Later extensions to the CAPM added additional risk factors and are known as multifactor CAPMs.

Capital asset pricing model

Financial analysts seek to understand the sources of variation in returns. One of their key tools is financial regression. Let the return of asset i at time t be r_{it}. Then a financial regression relates the asset's return to a risk-free rate r_t^f and independent variables.

$$r_{it} - r_t^f = \alpha_i + \beta_i(r_t^m - r_t^f) + \boldsymbol{\delta}'\mathbf{x}_t + e_{it}$$

The dependent variable is the time series of asset returns in excess of the risk-free rate, $r_{it} - r_t^f$. The key independent variable is the return on the market, also in excess of the risk-free rate, $(r_t^m - r_t^f)$. The parameter β_i is the coefficient on adjusted market returns, that is, the "market beta". The presence of additional independent variables $\mathbf{x}_t$ allows for other factors or controls with coefficients $\boldsymbol{\delta}$. The term α_i is an intercept in the regression and is of particular interest in financial statistics. Finally, e_{it} is a mean-zero error term. Such time-series regression models are known as CAPMs. These are time-series regressions because the estimate of β_i is driven by variation in a single stock over time.

In a financial regression, both the slope coefficient β_i and the intercept α_i are of interest. The slope coefficient measures the comovement of an asset with the market itself. An "aggressive" asset is one with $\beta_i > 1$, so that the asset moves more than one for one with market movements. A "conservative" asset has $0 < \beta_i < 1$; it moves in the same direction as the market but less than one for one. An asset with $\beta_i = 0$ has returns that are uncorrelated with market returns. Finally, an asset with $\beta_i < 0$ moves against the market, yielding positive returns during market downturns and vice versa; it serves as a hedge. The intercept α_i captures average returns that are not explained by the independent variables in the regression. Under the CAPM theory, the intercept should be zero. This forms the basis for a test of the theory.

So far, we have looked at a single asset. But we could fit one regression for each asset, allowing for a different intercept and slope coefficient for each asset. The resulting model is

$$\begin{aligned} r_{1t} - r_t^f &= \alpha_1 + \beta_1(r_t^m - r_t^f) + \boldsymbol{\delta}_1'\mathbf{x}_t + e_{1t} \\ \vdots \quad &= \quad \vdots \\ r_{Kt} - r_t^f &= \alpha_K + \beta_K(r_t^m - r_t^f) + \boldsymbol{\delta}_K'\mathbf{x}_t + e_{Kt} \end{aligned}$$

Notice that the adjustment variable r_t^f and the independent variables are common to all regressions. Letting $z_{it} = r_{it} - r_t^f$, $\mathbf{z}_t = (z_{1t}, \ldots z_{Kt})'$, and $f_t = r_t^m - r_t^f$, the collection of time-series regressions is a model,

$$\mathbf{z}_t = \boldsymbol{\alpha} + \boldsymbol{\beta} f_t + \boldsymbol{\Gamma}\mathbf{x}_t + \mathbf{e}_t$$

where $\boldsymbol{\alpha}$ is the $K \times 1$ vector of intercepts, $\boldsymbol{\beta}$ is the $K \times 1$ vector of slope coefficients, $\boldsymbol{\Gamma}$ is an $K \times C$ matrix of coefficients on the C independent control variables $\mathbf{x}_t$, and $\mathbf{e}_t$ is the $K \times 1$ vector of disturbance terms.

The CAPM is based on the assumption that the factors f_t explain all the variation in expected returns. This assumption generates two testable implications. First, additional factors beyond those included in one's theory should enter the regressions with coefficients of zero. Second, the intercepts $\boldsymbol{\alpha}$ should all be jointly zero. The latter implication is frequently assessed using the Gibbons–Ross–Shanken test.

In Stata, you can use `finregress capm` to fit a CAPM; see [FIN] **finregress capm**.

Fama–MacBeth regression

One result of the above time-series procedure is an estimate of the risk measure β_i for each asset. Different assets have different levels of risk, and a key implication of the model is that assets with higher expected returns should have higher risk. Thus, while the CAPM regressions above are fit to time-series data, the single β_i estimate for each asset has a cross-sectional implication. Ignoring covariates, the expectation of returns is

$$E(r_i) - E(r^f) = \beta_i[E(r^m) - E(r^f)]$$

This implication can be tested in the cross-section of returns. The cross-sectional model is

$$E(r_i) - E(r^f) = \gamma + \lambda\beta_i$$

The dependent variable is the expected excess return on asset i. The independent variable is asset i's beta risk, β_i. The parameter λ captures how expected returns vary with beta risk; it is called the price of risk. If all sources of risk are "priced in" (that is, if they are all accounted for in setting the price of the asset), then $\gamma = 0$.

Fama and MacBeth (1973) devised a two-stage method for estimating the cross-sectional relationship between expected return and market risk. In the first stage, a collection of K time-series regressions of asset returns on the market portfolio is performed, as in the CAPM regressions above. This stage yields K estimated β_i coefficients. In the second stage, a cross-sectional regression is run at each time point t ($t = 1, 2, \ldots, T$) with the cross-section of excess returns as the dependent variable and the market risk factors β_i as the independent variable. The result of the second stage is a time series of estimated intercepts and slopes, $\{\hat{\gamma}_t\}$ and $\{\hat{\lambda}_t\}$. The final parameter estimates are the time-series average of the intercept and slope,

$$\hat{\gamma} = \frac{1}{T}\sum_{t=1}^{T}\hat{\gamma}_t \qquad \hat{\lambda} = \frac{1}{T}\sum_{t=1}^{T}\hat{\lambda}_t$$

The single estimate $\hat{\lambda}$ measures the degree to which expected return rises for a unit increase in beta. Deviations from this line are the pricing errors for individual assets, the α_i in the time-series regression. And $\hat{\gamma}$ can be thought of as an average of the pricing errors; if the covariates in the model capture the mean of expected returns, then $\hat{\gamma}$ will equal zero. An estimate of $\hat{\gamma}$ substantially different from zero indicates the presence of residual variation in expected returns that is not captured by the model.

Fama and MacBeth (1973) estimate standard errors as the standard deviation of these time series of coefficients,

$$\sigma^2(\hat{\gamma}) = \frac{1}{T^2}\sum_{t=1}^{T}(\hat{\gamma}_t - \hat{\gamma})^2 \qquad \sigma^2(\hat{\lambda}) = \frac{1}{T^2}\sum_{t=1}^{T}(\hat{\lambda}_t - \hat{\lambda})^2$$

The Fama–MacBeth regression assumes the underlying returns are independently distributed through time but allows for correlation among returns within a time period. The regression does not account for the fact that the β_i are estimated. Shanken (1992) proposed a correction to the standard error calculation that accounts for this.

Cochrane (2005, 245–251) provides a detailed discussion of the Fama–MacBeth regression and its relationship to standard methods in panel-data estimation.

In Stata, you can use `finregress fmb` to fit a Fama–MacBeth regression; see [FIN] **finregress fmb**.

Time-series methods

Financial statisticians also use a number of time-series models to characterize, explain, and forecast financial data. This section provides an overview of models. This overview is not exhaustive. See [TS] **Time series** for a more comprehensive introduction to methods used in time-series analysis and Stata's tools for time-series data. This overview highlights some of the most widely used methods.

Autoregressive moving-average models

Autoregressive moving-average (ARMA) models are a flexible class of models for characterizing univariate time series. The arma model takes as its building block a noise process e_t to build a time-series y_t as the sum of its own past values and current and past values of the noise.

$$y_t = \phi_0 + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + e_t + \theta_1 e_{t-1} + \cdots + \theta_q e_{t-q}$$

This is called an ARMA(p, q) model. The autoregressive terms are the terms involving lags of y_t itself. The moving-average terms are the terms involving the current and lagged values of the disturbances e_t. The parameters are $(\phi_0, \phi_1, \ldots, \phi_p, \theta_1, \theta_2, \ldots, \theta_q)$. The choice of p and q is called the lag order of the ARMA model. Various criteria can be used to select the lag order. Visual plots of autocorrelations can be used to assess dependence. Formal tests that use information criteria to tradeoff model complexity and explanatory power are also employed to estimate the proper lag order.

The ARMA model is designed to characterize time series that are stationary. Stationary time series have a constant long-run mean and variance, meaning they do not exhibit trends. Some financial time series do exhibit trends; for example, the price of an asset might trend upward or downward over time. Even then, a time series could have an ARMA structure after differencing it one or more times to remove a trend. Such models are called autoregressive integrated moving-average (ARIMA) models and provide an additional layer of flexibility. In a financial context, although the price of an asset might exhibit trending behavior, the returns (percentage changes) of the price might be more stable. Then one would use an ARIMA model for the price but an ARMA model for the returns.

The ARIMA model is used to capture persistence and predictability in the conditional mean of a time-series process. An enduring topic in financial statistics is whether asset returns have any predictability at all; thus, test of an ARIMA structure against the null of no such structure takes on considerable importance.

Stata has a command to choose the lag order of an ARIMA model; see [TS] **arimasoc**. Stata also has a command to estimate the parameters of ARIMA models; see [TS] **arima**. For predictions and forecasting after ARIMA models, see [TS] **arima postestimation**.

Autoregressive conditional heteroskedasticity models

Stock returns show time-varying volatility: periods of high variability tend to cluster together, producing strings of periods of high volatility and strings of periods of low volatility. The ARMA model, however, cannot capture this feature of the data because, in an ARMA model, volatility is constant over time. The autoregressive conditional heteroskedasticity (ARCH) model extends the ARMA model by modeling time-varying dynamics in the conditional variance. Where an ARIMA model is designed to capture predictable movements in the mean of a time series (like the rate of return on an asset), an ARCH model is designed to capture predictable movements in the variance of the series (like the periods of high or low volatility in asset returns).

In an autoregressive model, the time-series process follows

$$
\begin{aligned}
y_t &= \phi_0 + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + e_t \\
e_t &\sim N(0, \sigma^2)
\end{aligned}
$$

where σ^2 is a constant. The ARCH model explains the variance process with autoregressive and moving-average terms:

$$
\begin{aligned}
e_t &\sim N(0, \sigma_t^2) \\
\sigma_t^2 &= \gamma_0 + \alpha_1 e_{t-1}^2 + \cdots + \beta_1 \sigma_{t-1}^2 + \cdots
\end{aligned}
$$

The three simplest models are the ARCH model of order 1,

$$\sigma_t^2 = \gamma_0 + \alpha_1 e_{t-1}^2$$

the generalized ARCH (GARCH) model of order 1,

$$\sigma_t^2 = \gamma_0 + \beta_1 \sigma_{t-1}^2$$

and the GARCH$(1,1)$ model, which combines both,

$$\sigma_t^2 = \gamma_0 + \alpha_1 e_{t-1}^2 + \beta_1 \sigma_{t-1}^2$$

The ARCH terms are the analogue in the variance model to moving-average terms in an ARIMA model. The GARCH terms are the analogue in the variance model to the autoregressive terms in an ARIMA model. In an ARIMA model, a shock to e_t leads the mean of the series to change, with persistence in that change captured in the autoregressive and moving-average terms. Similarly, in a GARCH model, a shock to e_t^2 leads the variance of the series to change, with persistence in that change captured in the ARCH and GARCH terms. The result of a GARCH model is a series whose variance shows clustering of periods of high and low volatility.

The parameters estimated in a GARCH model are of direct interest in that they capture persistence in the conditional variance process. A fitted GARCH model can also be used to predict the conditional variance, both in sample and out of sample.

GARCH models were developed, in large part, to assist in characterizing financial time series. Engle (1982) introduced the ARCH model, and Bollerslev (1986) introduced the GARCH model. Since then, a large family of models has been developed to generalize ARCH models in various directions. The `arch` command estimates the parameters of many members of the ARCH family of models; see [TS] **arch**.

Vector autoregressive models

Where ARIMA models capture time dependence in a single variable, vector autoregressive (VAR) models jointly estimate dependence across a collection of variables. Returns to one asset can be affected by past returns to a different asset, reflecting cross-return dynamic dependence. Or returns to one asset can be affected by contemporaneous returns to other assets, reflecting contemporaneous or static dependence. VAR models allow the estimation of both kinds of dependence.

A VAR model has the structure

$$\mathbf{y}_t = \mathbf{A}_1\mathbf{y}_{t-1} + \cdots + \mathbf{A}_p\mathbf{y}_{t-p} + \mathbf{u}_t$$
$$E(\mathbf{u}_t\mathbf{u}_t') = \mathbf{\Sigma}$$

In this model, the parameters in the $(\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_p)$ matrices capture dynamic dependence across variables (returns). The residual for return i is u_i; cross-sectional dependence is captured in the off-diagonal elements of the covariance matrix of the residuals, $\mathbf{\Sigma}$. For example, if multiple assets are in the same economic sector, then sector-specific news is likely to cause correlated movements across all the asset returns in that sector.

The `var` command estimates parameters of a VAR model; see [TS] **var**.

Vector error-correction models

Vector error-correction (VEC) models add additional structure to the dynamics of variables in a VAR model. As in an ARMA model, a VAR model assumes the time series under consideration are stationary, meaning they do not show trends. The VEC model extends the VAR model to allow for a common, joint trend among the series. In macroeconomics and finance, such models arise naturally for variables that have trending behavior but whose ratio is stationary. In macroeconomics, gross domestic product and its components have trends, but the share of each component in gross domestic product can be modeled as stationary. In finance, asset prices and dividends each have trends, but the price-dividend ratio can be modeled as stationary.

In such situations, a VEC model arises. For two variables, say, log prices p_t and log dividends d_t, the VEC model takes the form

$$\Delta p_t = \alpha_1(d_{t-1} + \beta p_{t-1}) + a_{11}\Delta p_{t-1} + a_{12}\Delta d_{t-1} + u_{p,t}$$
$$\Delta d_t = \alpha_2(d_{t-1} + \beta p_{t-1}) + a_{21}\Delta p_{t-1} + a_{22}\Delta d_{t-1} + u_{d,t}$$

In this two-variable example, the variables are expressed in growth rates (log differences). The parameters $(a_{11}, a_{12}, a_{21}, a_{22})$ are typical VAR coefficients. New is the presence of the lagged linear combination of the levels of the variables, $d_{t-1}+\beta p_{t-1}$. This is the cointegrating vector, the relationship around which the variables fluctuate. The parameter β characterizes the cointegrating relationship. Easiest to interpret is $\beta = -1$, in which case the cointegrating vector is $d_t - p_t$, so that the growth rates of prices and dividends react to the lagged price-dividend ratio. The reaction coefficients are captured by (α_1, α_2), which can differ across equations. These parameters are interpretable: if $\alpha_1 > 0$, for instance, then an unusually high dividend-price ratio forecasts higher future price growth. The magnitude of the parameters (α_1, α_2) captures the speed of convergence back to the cointegrating trend.

The vec command estimates the parameters of VEC models; see [TS] **vec**. Campbell and Shiller (1987) use similar techniques to study two financial phenomena: the price-dividend relationship and the short-long interest-rate spread. VEC models can also be employed with more than two time series, for example, in characterizing exchange rate dynamics across a collection of countries.

Multivariate GARCH models

Multivariate GARCH (MGARCH) models capture time-varying heteroskedasticity in a multivariate setting. As in a VAR model, multiple variables are modeled jointly. As in a GARCH model, both the conditional mean and conditional variance are allowed to vary over time.

A general form of the MGARCH model is

$$\mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \mathbf{u}_t$$
$$\mathbf{u}_t = \mathbf{H}_t^{1/2}\boldsymbol{\nu}_t$$

In the first equation, $\mathbf{y}_t$ is a collection of variables to be modeled, $\mathbf{x}_t$ represents a collection of independent variables that can include lags of $\mathbf{y}_t$, and $\mathbf{u}_t$ are disturbance terms. In the second equation, the disturbances are broken into two components: a time-varying component $\mathbf{H}_t$ and a collection of zero-mean, unit variance, independently and identically distributed shocks $\boldsymbol{\nu}_t$. The GARCH dynamics are embedded in the $\mathbf{H}_t$ matrix. The multivariate analogue of the GARCH$(1,1)$ model discussed previously is

$$\text{vech}(\mathbf{H}_t) = \mathbf{s} + \mathbf{A}\,\text{vech}(\mathbf{u}_{t-1}\mathbf{u}_{t-1}') + \mathbf{B}\,\text{vech}(\mathbf{H}_{t-1})$$

MGARCH models are not identified in their full generality. Different submodels have been proposed that make different choices for the tradeoff between model complexity and parsimony.

The diagonal vech model fits the diagonal elements of $(\mathbf{A}, \mathbf{B})$ above, allowing every element of $\mathbf{H}_t$ to follow a univariate GARCH process. The family of conditional correlation MGARCH models fits a parsimonious model for the conditional correlations.

In all cases, the MGARCH model allows for comovements in volatility that cannot be captured by univariate GARCH models, because the VAR model allows for comovements in the conditional mean that cannot be captured by univariate ARIMA models. The mgarch command fits four types of MGARCH models; see [TS] **mgarch** for details.

This entry has described a collection of tools used by statisticians and econometricians to analyze financial time series. The next entry describes a workflow in Stata for analyzing such data, using tools from the fin suite of commands as well as Stata's general time-series commands.

References

Black, F. 1972. Capital market equilibrium with restricted borrowing. *Journal of Business* 45: 444–455. https://doi.org/10.1086/295472.

Boffelli, S., and G. Urga. 2016. *Financial Econometrics Using Stata*. College Station, TX: Stata Press.

Bollerslev, T. 1986. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics* 31: 307–327. https://doi.org/10.1016/0304-4076(86)90063-1.

Campbell, J. Y. 2018. *Financial Decisions and Markets: A Course in Asset Pricing*. Princeton, NJ: Princeton University Press.

Campbell, J. Y., A. W. Lo, and A. C. MacKinlay. 1997. *The Econometrics of Financial Markets*. Princeton, NJ: Princeton University Press. https://doi.org/10.2307/j.ctt7skm5.

Campbell, J. Y., and R. J. Shiller. 1987. Cointegration and tests of present value models. *Journal of Political Economy* 95: 1062–1088. https://doi.org/10.1086/261502.

Cochrane, J. H. 2005. *Asset Pricing*. Rev. ed. Princeton, NJ: Princeton University Press.

Engle, R. F. 1982. Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica* 50: 987–1007. https://doi.org/10.2307/1912773.

Fama, E. F., and J. D. MacBeth. 1973. Risk, return, and equilibrium: Empirical tests. *Journal of Political Economy* 81: 607–636. https://doi.org/10.1086/260061.

Hurn, S., V. L. Martin, P. C. B. Phillips, and J. Yu. 2020. *Financial Econometric Modeling*. Oxford: Oxford University Press.

Jensen, M. C. 1968. The performance of mutual funds in the period 1945–1964. *Journal of Finance* 23: 389–416. https://doi.org/10.1111/j.1540-6261.1968.tb00815.x.

Lintner, J. 1965. The valuation of risk assets and the selection of risky investments in stock portfolios and capital budgets. *Review of Economics and Statistics* 47: 13–37. https://doi.org/10.2307/1924119.

Shanken, J. 1992. On the estimation of beta-pricing models. *Review of Financial Studies* 5: 1–33. https://doi.org/10.1093/rfs/5.1.1.

Sharpe, W. F. 1964. Capital asset prices: A theory of market equilibrium under conditions of risk. *Journal of Finance* 19: 425–442. https://doi.org/10.1111/j.1540-6261.1964.tb02865.x.

Also see

[FIN] **fin** — Commands for analysis of financial data+

fin — Commands for analysis of financial data[+]

[+]These features are part of StataNow.

Description Remarks and examples Also see

Description

This entry provides an overview of the commands for analyzing financial data and an example of a workflow that uses these commands. We demonstrate the use of the commands described in this manual, as well as commands available for analyzing time-series data generally, in a financial context.

Here is a list of the manual entries that provide useful additional information.

Financial statistics commands

[FIN] **finportfolio**	Financial portfolio selection[+]
[FIN] **finregress capm**	Capital asset pricing model (CAPM)[+]
[FIN] **finregress capm postestimation**	Postestimation tools for finregress capm[+]
[FIN] **finregress fmb**	Fama–MacBeth regression[+]
[FIN] **finregress fmb postestimation**	Postestimation tools for finregress fmb[+]
[FIN] **finreturns**	Generate financial returns[+]
[FIN] **finsummarize**	Financial summary statistics[+]
[FIN] **finvalrisk**	Value at risk[+]

Date and time management

[TS] **tsset**	Declare data to be time-series data
[D] **bcal**	Business calendar file manipulation[+]
[D] **Datetime business calendars**	Business calendars
[D] **Datetime business calendars creation**	Business calendars creation

Unit-root tests

[TS] **dfgls**	DF-GLS unit-root test
[TS] **dfuller**	Augmented Dickey–Fuller unit-root test
[TS] **pperron**	Phillips–Perron unit-root test
[XT] **xtunitroot**	Panel-data unit-root tests

Time-series smoothers

[TS] **tsfilter**	Filter a time series for cyclical components
[TS] **tssmooth**	Smooth and forecast univariate time-series data

Factor models and PCA

[TS] **dfactor**	Dynamic-factor models
[MV] **factor**	Factor analysis
[MV] **pca**	Principal component analysis

Time-series commands

[TS] **arch**	Autoregressive conditional heteroskedasticity (ARCH) family of estimators
[TS] **arima**	ARIMA, ARMAX, and other dynamic regression models
[TS] **arimasoc**	Obtain lag-order selection statistics for ARMAs
[TS] **ivlpirf**	Instrumental-variables local-projection impulse–response functions
[TS] **lpirf**	Local-projection impulse–response functions
[TS] **mgarch**	Multivariate GARCH models
[TS] **mgarch ccc**	Constant conditional correlation multivariate GARCH model
[TS] **mgarch dcc**	Dynamic conditional correlation multivariate GARCH model
[TS] **mgarch dvech**	Diagonal vech multivariate GARCH model
[TS] **mgarch vcc**	Varying conditional correlation multivariate GARCH model
[TS] **mswitch**	Markov-switching regression models
[TS] **newey**	Regression with Newey–West standard errors
[TS] **threshold**	Threshold regression
[TS] **var**	Vector autoregressive models
[TS] **var svar**	Structural vector autoregressive models
[TS] **var ivsvar**	Instrumental-variables structural vector autoregressive models
[TS] **vec**	Vector error-correction models

Panel-data commands

[XT] **xtreg**	Linear models for panel data+
[XT] **xtivreg**	Instrumental variables and two-stage least squares for panel-data models
[XT] **xtvar**	Panel-data vector autoregressive models

Remarks and examples

Remarks are presented under the following headings:

Returns and data management

Throughout this introduction, we use a running example consisting of data on the stock prices of 25 fictional companies, along with data on a stock-market index and a risk-free rate of return. Our data consist of 780 monthly observations containing the end-of-month values for these stock prices.

Let's open the data and describe the first few variables.

```
. use https://www.stata-press.com/data/r19/finex
(Fictional stock price data)
. describe datestr-wgt
```

Variable name	Storage type	Display format	Value label	Variable label
datestr	str11	%11s		String date
datem	int	%tm		Monthly date
sp500	double	%10.0g		S&P 500
vol	float	%9.0g		Volatility index
fedfunds	float	%9.0g		Federal funds rate
acme	float	%9.0g		Aciron Medical, Inc.
bat	float	%9.0g		Boron Advanced Technologies
iron	float	%9.0g		Industrial Operations Network
dune	float	%9.0g		Digital Urban Network Enterprise
tyr	float	%9.0g		Tyndale Resources Group
glo	float	%9.0g		Green Logistics, Inc.
spa	float	%9.0g		Space Rocket MFG
wgt	float	%9.0g		Widget Gadgets

The first two variables contain string and numeric dates. The latter is used to tell Stata about the time-series structure of the data. The third variable, `sp500`, contains a stock-market index. The variable `fedfunds` contains the annualized federal funds rate, which we take as the risk-free rate of return. The remaining variables are the stock prices of fictional firms.

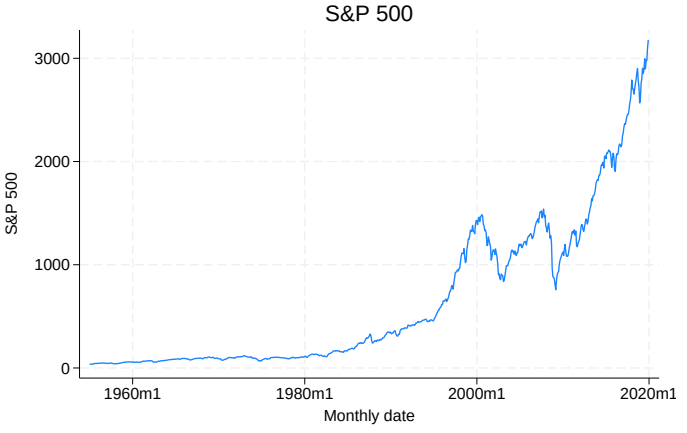
The `tsreport` command confirms that the dataset has been preset with a time-series structure and provides some information about the date variable. See [TS] **tsreport** for more on understanding the structure of your time-series data.

```
. tsreport
Time variable: datem
-----------------------------
Starting period =  1955m1
Ending period   = 2019m12
Number of obs   =     780
Number of gaps  =       0
```

The monthly date variable ranges from 1955 through 2019 without gaps, for a total of 780 observations. We are starting with a clean dataset, with minimal data processing needed and all the time-series issues (gaps, missing values, etc.) already resolved. The data you use may not come in so neat a format, necessitating data management. In addition, there are special considerations needed for using daily data. We will address some of these considerations in *Date and time management* below.

We begin our analysis by looking at one of the series, say, the stock-market index. We use the `tsline` command as follows:

```
. tsline sp500, title("S&P 500")
```



The index rises over time, albeit with periodic substantial declines. One of our goals is to analyze the behavior of the market price. A second goal is to understand how the behavior of individual assets and portfolios of assets are related to the behavior of the overall market.

The return on an asset price is a measure of how the asset price changes over time. There are various types of returns. The most fundamental is the simple return, defined as the proportional change in the asset price each period. Other concepts of return include logarithmic returns, annualized returns, and seasonalized returns.

The `finreturns` command computes returns for one or more variables. To compute the simple returns for the stock-market index, we type

```
. finreturns sp500, simple(rmkt) multiply(100)
(all returns multiplied by 100)
Simple returns generated for variable sp500:
```

	datem	sp500	rmkt
1.	1955m1	35.6	.
2.	1955m2	36.79	3.3426966
3.	1955m3	36.5	-.78825768
4.	1955m4	37.76	3.4520548
5.	1955m5	37.6	-.42372881
6.	1955m6	39.78	5.7978723
7.	1955m7	42.69	7.3152338
8.	1955m8	42.43	-.60904193
9.	1955m9	44.34	4.5015319
10.	1955m10	42.11	-5.0293189

We specify that we want to compute returns for `sp500`. The `simple()` option specifies that we want the simple return, and the `rmkt` variable is a new variable containing the return we calculated. The `multiply()` option multiplies the result by the number specified, in this case, 100, so that the result can be interpreted as percentage change from the previous month. A preview of the computed returns

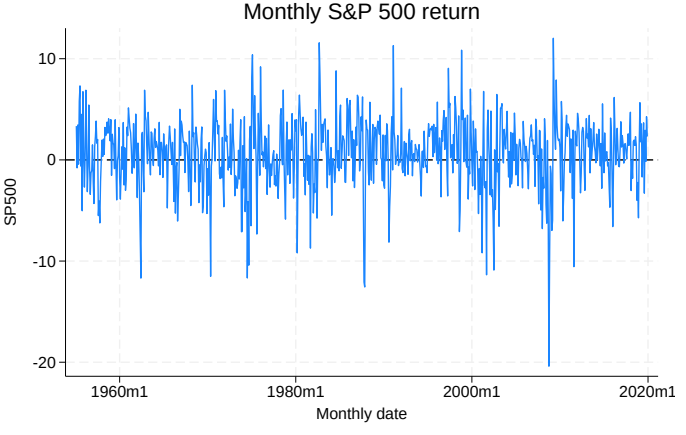
is shown, indicating the date, the original variable, and the new return variable. In observation 2, for example, the simple return 3.34 is equal to $100 \times (36.79/35.6 - 1)$ and indicates a 3.34% increase over the previous month.

```
. summarize rmkt
```

Variable	Obs	Mean	Std. dev.	Min	Max
rmkt	779	.6387889	3.458349	-20.39114	12.02171

The average monthly return is 0.64%, slightly more than one-half of a percent. The standard deviation of these returns is large, about 3.46, with some months being disasters (a minimum monthly return of -20%) and some months showing strong positive returns (a maximum one-month gain of 12%). We can graph the monthly returns:

```
. tsline rmkt, yline(0) title("Monthly S&P 500 return")
```



The pattern of returns shows a mild degree of persistence: months with higher than typical returns are followed by months that also have higher than typical returns. More visually apparent is the clustering of volatility. There are periods where the variance of returns expands and periods of relative calm where the variance is lower. The question of whether financial time series are predictable and the question of characterizing their volatility have been longstanding topics for financial statistics. We explore the predictability of returns and the nature of the time-varying volatility below.

In addition to the market return, we are also interested in the returns on the stock prices of our collection of firms. We use `finreturns` again, this time with the full set of stock prices. Instead of typing all 25 firm names, we use the varlist range notation *varname1*`-`*varnameK* to specify every variable between the two named variables in the dataset, inclusive.

```
. finreturns acme-tks, simple(r_) multiply(100) nopreview
(all returns multiplied by 100)

Simple returns generated for variables acme, bat, iron, dune, tyr, glo,
spa, wgt, bar, yum, aaa, afh, ard, cph, das, dil, ege, epg, goa, jml, khc,
krg, kth, nhb, and tks.
```

Three aspects of the results are worth noting. First, because there was more than one asset specified, the name `r_` is interpreted as a stub; the resulting new variables are `r_acme`, `r_bat`, and so on, up to `r_tks`. The return variables are stored in order in the dataset, so it is easy to use the varlist range notation in subsequent commands. Second, we specified `nopreview` to suppress the preview of the results. By

default, a preview of the returns for all 25 assets would be displayed. Third, if we `describe` the returns, we see that each return has been labeled with the name of the variable it was derived from in capital letters.

```
. describe r_acme-r_tyr
Variable      Storage   Display    Value
    name         type    format    label      Variable label
------------------------------------------------------------------------------
r_acme          double  %10.0g                ACME
r_bat           double  %10.0g                BAT
r_iron          double  %10.0g                IRON
r_dune          double  %10.0g                DUNE
r_tyr           double  %10.0g                TYR
```

These automatic labels make it easy to keep track of how a given return was generated.

We can summarize the characteristics of the returns using the `finsummarize` command. To save space, we summarize only the returns for the first five assets:

```
. finsummarize r_acme-r_tyr
Financial summary statistics
Number of obs = 779
Sample: 1955m2 thru 2019m12
```

	Mean	Std. dev.	Sharpe ratio
ACME	0.4517	1.1856	0.3810
BAT	0.9052	5.4073	0.1674
IRON	1.0214	6.5625	0.1556
DUNE	1.0593	6.6088	0.1603
TYR	0.7028	3.9642	0.1773

By default, the mean, standard deviation, and Sharpe ratio of each return is displayed. The Sharpe ratio is the ratio of the mean (possibly adjusted for a risk-free rate) over the standard deviation. Higher values indicate assets with higher returns, lower volatility, or a combination of the two. As before, these are monthly returns in percentages. The mean return varies across assets, from 0.45% to 1.06%, with a ratio of about 2. One goal of asset pricing is to understand the variation in average returns and see whether any other variables can predict which kinds of assets will show high returns. Also notice that the standard deviation seems to rise as mean returns rise.

Portfolio construction

Portfolios are linear combinations of assets. Assets can be combined to reduce variability by diversifying one's investments. The `finportfolio` command produces portfolios based on various criteria. In this section, we build 4 portfolios from the 25 asset returns.

The first portfolio we create will be the simplest one: an equally weighted portfolio of all 25 stocks. This is accomplished with `finportfolio equal`. We specify the `generate()` option to generate the `p1` variable with the portfolio returns. We will use this portfolio return variable later, comparing its performance with the individual stocks and with other portfolios.

```
. finportfolio equal r_acme-r_tks, generate(p1, label("Equal"))
Equally weighted portfolio
Number of obs = 779
Sample: 1955m2 thru 2019m12
```

	Weight
ACME	.04
BAT	.04
IRON	.04
DUNE	.04
TYR	.04
GLO	.04
SPA	.04
WGT	.04
BAR	.04
YUM	.04
AAA	.04
AFH	.04
ARD	.04
CPH	.04
DAS	.04
DIL	.04
EGE	.04
EPG	.04
GOA	.04
JML	.04
KHC	.04
KRG	.04
KTH	.04
NHB	.04
TKS	.04

```
Portfolio return    = 0.6697
Portfolio std. dev. = 2.7483
Risk-free rate      = 0.0000
Sharpe ratio        = 0.2437
```

The main table provides the weights associated with each return. The equally weighted portfolio does not have much going on; each weight is 1/25, equal to a 4% share of the portfolio. Below the table of weights is some information about the portfolio performance. We have the portfolio mean and standard deviation; if a risk-free rate was specified, its return is also displayed. Then the Sharpe ratio is displayed. We see that the equally weighted portfolio returns 0.67% per month with a standard deviation of 2.75%.

Now, a savvy investor might not be satisfied with an equally weighted portfolio. Instead, an investor might want to craft a portfolio that met certain criteria, such as the lowest overall possible variance or the lowest variance while still hitting a target return or maximizing the reward-to-risk ratio (the Sharpe ratio). The minimum variance portfolio chooses the combination of weights that delivers the lowest variance. The minimum variance portfolio can be chosen to meet the global minimum or to minimize variance for a specified, target return. Let's specify `target(0.67)` to match the monthly return on the equally weighted portfolio and specify `generate(p2)` to store this new portfolio's return in a new variable.

```
. finportfolio minvariance r_acme-r_tks, target(0.67)
> generate(p2, label("MinVar"))
Minimum variance portfolio with fixed return
Short selling allowed
Number of obs = 779
Sample: 1955m2 thru 2019m12
```

	Weight
ACME	.0234078
BAT	.1227878
IRON	.0562631
DUNE	.145313
TYR	-.2108211
GLO	.0593816
SPA	.1825036
WGT	-.0500623
BAR	-.0048355
YUM	.0970868
AAA	.1528412
AFH	-.067757
ARD	.0436591
CPH	-.1453048
DAS	.0166238
DIL	.1613341
EGE	-.112362
EPG	.0370243
GOA	.5820644
JML	-.0728716
KHC	.0665349
KRG	-.1612624
KTH	-.081871
NHB	.0315904
TKS	.1287319

```
Portfolio return    = 0.6700 (fixed return)
Portfolio std. dev. = 1.0311
Risk-free rate      = 0.0000
Sharpe ratio        = 0.6498
```

The asset weights now vary in the combination that delivers the lowest portfolio variance for the specified return. Some of the weights are negative, implying a short position in those assets. Below the weight table, we see that the minimum variance portfolio delivers the same 0.67% monthly return as the equally weighted portfolio but does so with less than half of the standard deviation: 1.03% instead of 2.75%.

One might wish to construct portfolios that permit only positive weights on all assets; these are no-short portfolios. `finportfolio` allows this constraint with the `noshort` option. We run the same exercise, choosing the minimum variance portfolio subject to the restriction of no short sales.

```
. finportfolio minvariance r_acme-r_tks, target(0.67) noshort
> generate(p3, label("NoShort"))
Iteration 0:  Penalized variance =  7.5745874
Iteration 1:  Penalized variance =  6.0526077
Iteration 2:  Penalized variance =  5.7479174
Iteration 3:  Penalized variance =  5.4642297
Iteration 4:  Penalized variance =  5.2008879
Iteration 5:  Penalized variance =  4.9567827
Iteration 6:  Penalized variance =  4.7298329
Iteration 7:  Penalized variance =  4.3146947
Iteration 8:  Penalized variance =  3.9610262
Iteration 9:  Penalized variance =  3.6530792
  (iteration log omitted)
Iteration 15: Penalized variance =  2.0569964
Iteration 16: Penalized variance =   1.821173
Iteration 17: Penalized variance =  1.6890968
Iteration 18: Penalized variance =  1.6151142
Iteration 19: Penalized variance =   1.544494
Iteration 20: Penalized variance =    1.52092
Iteration 21: Penalized variance =  1.5208523
Iteration 22: Penalized variance =  1.5208441
Iteration 23: Penalized variance =  1.5208416
Iteration 24: Penalized variance =  1.5208415

Minimum variance portfolio with fixed return
Short selling not allowed

Number of obs = 779
Sample: 1955m2 thru 2019m12
```

	Weight
ACME	.0001638
BAT	.0443116
IRON	.0007996
DUNE	.1363128
TYR	.0000514
GLO	.0001381
SPA	.0572409
WGT	.000116
BAR	.0001778
YUM	.0254636
AAA	.0390746
AFH	.0000705
ARD	.0104981
CPH	.0000717
DAS	.0001209
DIL	.0863905
EGE	.0000611
EPG	.0001493
GOA	.5966434
JML	.0000725
KHC	.0005557
KRG	.0000507
KTH	.0000748
NHB	.0005682
TKS	.0008225

```
Portfolio return    = 0.6700 (fixed return)
Portfolio std. dev. = 1.2285
Risk-free rate      = 0.0000
Sharpe ratio        = 0.5454
```

One new aspect of the output is the iteration log. Minimizing variance subject to a no-short constraint is a nonlinear problem that requires an iterative approach to solve. The iteration log displays progress toward the goal, minimum variance. We can see that, at each step, the variance of the portfolio is reduced. The process stops when the weights are found that deliver as small a variance as possible, subject to the constraints. The resulting weights are all positive, as intended. By forcing all weights to be positive, we see that relative weights within the portfolio have shifted somewhat. Notice also that the no-short requirement has increased the portfolio standard deviation somewhat, from 1.03% to 1.23%.

As a fourth portfolio, we construct the portfolio with the highest ratio of return to standard deviation. This is the maximum Sharpe ratio portfolio. It does not have a target return; instead it chooses the target return to deliver the best return-to-risk ratio.

```
. finportfolio maxsharpe r_acme-r_tks, noshort gamma(0.001)
> generate(p4, label("MaxSharpe"))
Iteration 0:  Penalized Sharpe ratio =  .16318784  (not concave)
Iteration 1:  Penalized Sharpe ratio =  .19263253  (not concave)
Iteration 2:  Penalized Sharpe ratio =   .2746519  (not concave)
Iteration 3:  Penalized Sharpe ratio =  .32067251  (not concave)
Iteration 4:  Penalized Sharpe ratio =  .33244638  (not concave)
Iteration 5:  Penalized Sharpe ratio =   .3513766  (not concave)
Iteration 6:  Penalized Sharpe ratio =  .38394785  (not concave)
Iteration 7:  Penalized Sharpe ratio =  .44470668  (not concave)
Iteration 8:  Penalized Sharpe ratio =  .51944305  (not concave)
Iteration 9:  Penalized Sharpe ratio =  .55928097  (not concave)
Iteration 10: Penalized Sharpe ratio =  .57315047  (not concave)
Iteration 11: Penalized Sharpe ratio =  .58966733  (not concave)
Iteration 12: Penalized Sharpe ratio =   .5922082  (not concave)
Iteration 13: Penalized Sharpe ratio =  .59788206  (not concave)
Iteration 14: Penalized Sharpe ratio =  .64981786
Iteration 15: Penalized Sharpe ratio =  .71444962
Iteration 16: Penalized Sharpe ratio =  .74339013
Iteration 17: Penalized Sharpe ratio =  .75825625
Iteration 18: Penalized Sharpe ratio =  .76269832
Iteration 19: Penalized Sharpe ratio =  .76439582
Iteration 20: Penalized Sharpe ratio =  .76507737
Iteration 21: Penalized Sharpe ratio =  .76516526
Iteration 22: Penalized Sharpe ratio =  .76516566
Iteration 23: Penalized Sharpe ratio =  .76516566
```

```
Maximum Sharpe ratio portfolio
Short selling not allowed
Number of obs = 779
Sample: 1955m2 thru 2019m12
```

	Weight
ACME	.1005565
BAT	.0016509
IRON	.0010381
DUNE	.0011628
TYR	.0018101
GLO	.0079593
SPA	.1879825
WGT	.0034215
BAR	.0045021
YUM	.0206193
AAA	.0575591
AFH	.0045532
ARD	.1706966
CPH	.0323272
DAS	.0153331
DIL	.0069934
EGE	.0015328
EPG	.0016666
GOA	.281921
JML	.0176177
KHC	.0020375
KRG	.0015561
KTH	.0033547
NHB	.0681455
TKS	.0040023

```
Portfolio return    = 0.4817
Portfolio std. dev. = 0.5458
Risk-free rate      = 0.0000
Sharpe ratio        = 0.8826
```

The Sharpe ratio jumps from 0.55 in the previous minimum variance portfolio to 0.88 in the maximum Sharpe ratio portfolio. This return-to-risk ratio is achieved with a lower monthly return, 0.48% instead of 0.67%, but a much lower standard deviation, 0.55% versus 1.03% to 2.75% in the case of the minimum variance and equally weighted portfolios, respectively.

Portfolio summaries

Next we continue to explore the properties of the portfolios we have constructed. Again, we use `finsummarize`.

```
. finsummarize p*
Financial summary statistics
Number of obs = 779
Sample: 1955m2 thru 2019m12
```

	Mean	Std. dev.	Sharpe ratio
Equal	0.6697	2.7483	0.2437
MinVar	0.6700	1.0311	0.6498
NoShort	0.6700	1.2285	0.5454
MaxSharpe	0.4817	0.5458	0.8826

We use the notation `p*` to indicate all variables in the dataset that begin with `p` and have any additional text after this first letter. The characteristics of the portfolios are displayed, repeating what we have already seen from `finportfolio` but now in a more easy-to-read format. Adding the S&P 500 market return `rmkt` in the `benchmark()` option allows `finsummarize` to compute more information.

```
. finsummarize p*, benchmark(rmkt)
Financial summary statistics
Number of obs = 779
Sample: 1955m2 thru 2019m12
```

	Mean	Std. dev.	Sharpe ratio	Treynor index	Jensen's alpha	beta
Equal	0.6697	2.7483	0.2437	0.8563	0.1701	0.7820
MinVar	0.6700	1.0311	0.6498	7.8706	0.6156	0.0851
NoShort	0.6700	1.2285	0.5454	4.9402	0.5834	0.1356
MaxSharpe	0.4817	0.5458	0.8826	12.0049	0.4561	0.0401

Note: Variable **rmkt** used as benchmark asset.

Adding a market index allows us to compute the Treynor index, Jensen's alpha, and market beta. The Treynor index, like the Sharpe ratio, is a return-to-risk measure. Jensen's alpha is the intercept in a regression of the asset return on the market return. Positive values of Jensen's alpha indicate returns to the portfolio that are attained over and above those returns derived from the asset's association with the market. Market beta is the covariance of the portfolio return with the market. Higher values indicate an asset or a portfolio that covaries more strongly with the market. The Treynor index is a function of the market beta; it divides the return by market beta. Higher values of Treynor's index indicate the existence of higher returns that are achieved without also exposing the investment to additional market risk.

The equally weighted portfolio has the highest standard deviation at 2.75%, the lowest Sharpe ratio at 0.24, and much higher exposure to market risk than the other portfolios with a beta of 0.78. Hence, the equally weighted portfolio moves closely with the market. By contrast, both of the minimum variance portfolios (`MinVar` and `NoShort`) attain a much lower standard deviation, at 1.03% and 1.23%, respectively. They thus attain the same monthly return at half the monthly volatility. Both minimum variance portfolios have relatively high alpha, of about 0.60% monthly. This may be interpreted as these assets

yielding about 0.60% returns per month in a way that is uncorrelated with market risk. Their market betas are quite low, 0.09 and 0.14, respectively, indicating that the portfolio weights in the minimum variance portfolios shield them from market risk.

The maximum Sharpe ratio portfolio has a lower standard deviation than even the minimum variance portfolios, at 0.55%. This is possible because the maximum Sharpe ratio portfolio trades off some average return: 0.48% per month instead of 0.67% per month. This portfolio is nearly unrelated to the market, with a beta of 0.04. Thus, the maximum Sharpe ratio portfolio trades off lower return for lower variance, maximizing the return-to-risk ratio. This portfolio also achieves its returns in a way that is almost totally insulated from market fluctuations.

Financial regressions

In constructing models for returns, log returns are a more convenient object than simple returns. Log returns approximate simple returns when returns are small, and they have the advantage of being symmetric. For example, a 100% increase in log return followed by a 100% decrease in log return brings the asset back to its initial value. This is not true for simple returns, where a 100% increase would have to be followed by a 50% decrease to land back at the initial value. Log returns can thus be added and averaged arithmetically. This is useful in a modeling setting, where the statistical properties of the model are understood as the behavior of averages of the time series.

We use `finreturns` again to create variables holding the log return on the market and the log return on the stocks we want to analyze. We multiply log returns by 100 so that they approximate percentage changes.

```
. finreturns sp500, log(lnr_mkt) multiply(100)
(all returns multiplied by 100)
Log returns generated for variable sp500:
```

	datem	sp500	lnr_mkt
1.	1955m1	35.6	.
2.	1955m2	36.79	3.2880431
3.	1955m3	36.5	-.79138085
4.	1955m4	37.76	3.3938081
5.	1955m5	37.6	-.42462909
6.	1955m6	39.78	5.6360223
7.	1955m7	42.69	7.0600427
8.	1955m8	42.43	-.61090416
9.	1955m9	44.34	4.4031545
10.	1955m10	42.11	-5.1601962

```
. finreturns acme-tks, log(lnr_) multiply(100) nopreview
(all returns multiplied by 100)
Log returns generated for variables acme, bat, iron, dune, tyr, glo, spa,
wgt, bar, yum, aaa, afh, ard, cph, das, dil, ege, epg, goa, jml, khc, krg,
kth, nhb, and tks.
```

The `lnr_mkt` variable now holds the log market return. We have 25 variables containing the log returns for each stock, `lnr_acme` through `lnr_tks`.

We will also need a monthly risk-free rate of return for our analysis. We generate it using double precision to achieve more accuracy when we use it for further computations. To construct it, we type

```
. generate double rf = 100 * ln(1+fedfunds/100)/12
```

This construction does two things. First, it converts the `fedfunds` interest rate, measured in annual percentage change, into a log return equivalent to the log returns used for the other assets. Second, dividing the log return by 12 gives a monthly equivalent to the annual return. The result is the short-term federal interest rate at a monthly frequency. We will use this risk-free rate to compute excess returns on our stocks, among other things.

Capital asset pricing model

The capital asset pricing model (CAPM) relates an asset's return to the market return, after adjusting for the risk-free rate. Apart from the market return, other characteristics or "factors" describing the economy, the market, the industry, and the asset itself can be included as independent variables in the model to serve as controls. The return of asset i at time t, net of the risk-free rate, is $r_{it} - r_t^f$. The market excess return is $r_t^m - r_t^f$. Without controls, the linear regression relating these quantities is

$$r_{it} - r_t^f = \alpha_i + \beta_i(r_t^m - r_t^f) + e_{it}$$

Each asset has its own asset-specific intercept α_i and slope coefficient β_i. Both the slope coefficient and the intercept have financial interpretation. The slope coefficient measures how much the asset's excess return is expected to rise when the market excess return rises by one unit. The intercept captures the average excess return of the asset when the market excess return is zero. Hence, α_i measures the average return on the asset that is independent of the market return. `finregress capm` fits this regression for a set of assets. We now use it to compute intercepts and slope coefficients for the first five log returns we obtained earlier.

```
. finregress capm lnr_acme-lnr_tyr = lnr_mkt, rfrate(rf) adjust
Capital asset pricing model
Sample: 1955m2 thru 2019m12                          Number of obs = 779
```

	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
lnr_acme						
lnr_mkt	.1316993	.0102147	12.89	0.000	.111679	.1517197
_cons	.0322367	.0378174	0.85	0.394	-.041884	.1063575
lnr_bat						
lnr_mkt	1.530731	.0114377	133.83	0.000	1.508314	1.553148
_cons	.0760026	.0416309	1.83	0.068	-.0055925	.1575977
lnr_iron						
lnr_mkt	1.871524	.012273	152.49	0.000	1.847469	1.895579
_cons	.0556327	.0454103	1.23	0.221	-.0333698	.1446352
lnr_dune						
lnr_mkt	1.878887	.0123726	151.86	0.000	1.854637	1.903137
_cons	.0895071	.046235	1.94	0.053	-.0011119	.1801261
lnr_tyr						
lnr_mkt	1.087576	.0117232	92.77	0.000	1.064599	1.110553
_cons	.0282078	.0443328	0.64	0.525	-.0586828	.1150984

```
Notes: Dependent variables adjusted for risk-free rate rf.
       Independent variable lnr_mkt adjusted for risk-free rate rf.
```

The basic syntax is *asset returns* = *independent variables*. When a risk-free rate is specified in `rfrate()`, the asset returns on the left-hand side are adjusted by the risk-free rate. However, the independent variables (the factors) on the right-hand side are not. This is because the independent variables may already be excess returns or may be a factor that does not need to be adjusted. To indicate that an independent variable needs to be adjusted by the risk-free rate, specify the `adjust()` option.

The CAPM results here show variation in the slope coefficients and in the intercepts. These slope coefficients have the standard interpretation of regression coefficients. Because we have used log returns and adjusted for the risk-free rate, our dependent variables are excess log returns for the five assets we specified (ACME, BAT, IRON, DUNE, and TYR). Similarly, the independent variable is the excess log market return. We can interpret the estimates in these units. As the excess log market return rises by one unit, the expected excess log return for ACME rises by 0.13, so we expect excess log returns for ACME to rise less than those for the market. Expected excess log returns for BAT, IRON, and DUNE all rise more than one for one with excess log market returns. We expect excess log returns for these assets to be up (between 1.53 and 1.88). Finally, the expected excess log return for TYR has a slope coefficient of 1.09, so it moves approximately one for one with the excess log market return. These coefficients are also called an asset's market exposure.

Under a strict interpretation of the CAPM, all variation in asset returns should be driven by the independent variables. If this is true, then all the intercepts will be zero. A key test of the theory, then, is a test for whether all the intercepts are jointly zero. The Gibbons–Ross–Shanken test performs this test. We use `estat grstest` to perform the test and specify the `finite` option to obtain a finite-sample F test.

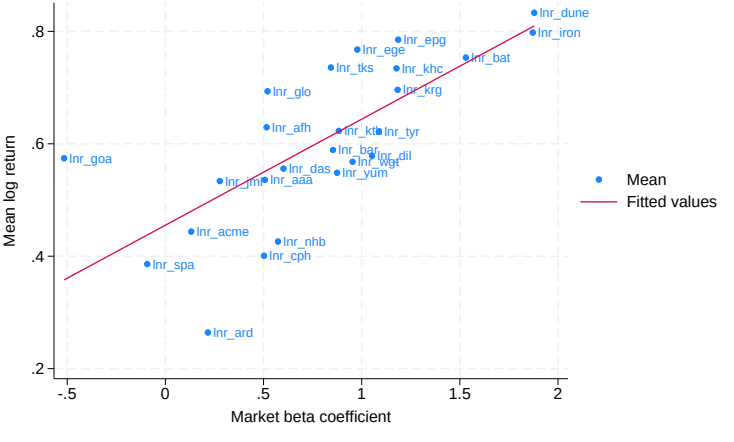
```
. estat grstest, finite
Gibbons-Ross-Shanken test
H0: All intercept terms are zero
  No. of dependent vars. =       5
No. of independent vars. =       1
     No. of time periods =     779
                 F(5, 773) =  1.321
                  Prob > F = 0.2532
```

For these five test assets, we cannot reject the null that the intercepts are jointly zero. The interpretation is that the specified factors, in this case the sole factor being the market return, adequately explain the pattern of average returns across assets.

Fama–MacBeth regression

While the time-series regressions above capture variation in returns over time, many questions in asset pricing revolve around the cross-section of average (or expected) returns. Why does one asset return more than another, consistently? One answer is that the higher-returning factor bears more risk, for which the asset holder is "paid" by also offering a higher average return. In the time-series regressions above, an asset's risk is measured by its covariance with the market. Riskier assets move with the market. By contrast, an asset that does not move with the market pays off precisely when the market does not, which is an advantage to a risk-adverse investor. A risk-adverse investor would hold highly risky assets only if those assets also paid higher average returns. This is the key behind the next regressions.

The Fama–MacBeth regression begins with the coefficients estimated in the time-series CAPM and then runs a second-stage regression of average returns on the first-stage coefficients. Visually, the regression being run is the following:



The intuition is that assets that are riskier (have higher beta) should also have higher returns. The slope of this line captures the rise in average return resulting from taking on one more "unit" of risk, where a unit of risk is measured by the responsiveness of the asset to the market.

But simply running this regression on the averages is unwise for two reasons. First, by taking averages, the regression understates the variability in the second-stage estimate. Second, the betas themselves are estimated, and a naive regression understates the variability of the second-stage estimate by not accounting for the variability of the first-stage estimates.

The Fama–MacBeth regression addresses the first problem by running this regression T times, once per time period, on the cross-section of returns. The final estimate is the average of the T cross-sectional estimates, and its variability is just the standard deviation of the cross-sectional estimates.

```
. finregress fmb lnr_acme-lnr_tks = lnr_mkt, rfrate(rf) adjust
Fama-MacBeth regression
Sample: 1955m2 thru 2019m12                          Number of obs     = 779
Method: OLS                                          Number of depvars =  25
```

depvar_means	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
beta						
lnr_mkt	.1886384	.1270712	1.48	0.138	-.0604165	.4376934
_cons	.06874	.0159916	4.30	0.000	.0373971	.1000829

```
Dependent variables: lnr_acme lnr_bat lnr_iron lnr_dune lnr_tyr lnr_glo
                     lnr_spa lnr_wgt lnr_bar lnr_yum lnr_aaa ... lnr_tks
Notes: Dependent variables adjusted for risk-free rate rf.
       Independent variable lnr_mkt adjusted for risk-free rate rf.
```

We have log returns for 25 assets used as dependent variables in the first stage. For brevity, the command does not list all dependent variables below the table, but you can click on the dots to see the full list. Because we specified the `rfrate(rf)` and `adjust` options, the first-stage models regress excess log returns for these assets on excess log market return.

The first coefficient reported is the slope coefficient. It is equivalent to the slope of a regression of mean returns on beta risk. The dependent variable is the cross-section of mean returns, denoted here by `depvar_means`. The independent variable is beta risk of `lnr_mkt`, denoted as `lnr_mkt` in the `beta` equation, which was estimated in the first-stage regressions.

In our example, `depvar_means` refers to the average excess log returns of the listed dependent variables. The coefficient `lnr_mkt` captures the increase in expected average excess log return for accepting one more unit of beta risk. This slope is often called the price-of-risk coefficient. The coefficient `_cons` is the constant term. It can be interpreted as the average excess log return on an asset with zero beta risk.

The average price-of-risk coefficient for `lnr_mkt` is positive, indicating that as market beta risk rises, expected return rises. For a one-unit increase in beta risk for excess log market returns, the expected average log return of considered assets would rise by about 0.19. For relatively small return values, the average price-of-risk coefficient estimated based on log returns can approximate the coefficient based on simple returns. This approximation holds in our example and, more generally, with daily and monthly data. So we could equivalently state that for a one-unit increase in market beta risk for excess simple returns, we expect average excess simple returns to rise by about 0.19. Thus, as investors take on more risk, on average, their returns also rise. However, the p-value on the coefficient of `lnr_mkt` suggests that its effect is not different from zero. A coefficient of exactly zero would indicate that investors are not rewarded for taking on additional risk. A negative coefficient would indicate that as exposure to the factor rises, average returns fall, indicating a "negative" average return for exposure to that factor.

Value at risk

We now turn to risk management. Summary statistics can provide measures of the average performance of an asset or portfolio, such as the mean return, and can provide summary measures of dispersion, such as the standard deviation. However, standard deviation is just one measure of dispersion. Also useful to know are other characteristics of a return's distribution, such as the so-called value at risk (VaR).

VaR is a quantity representing the potential loss on the value of an asset or a portfolio over a specified length of time for a given confidence level. If the VaR for an asset is \$300 at a 95% confidence level at the one-month horizon, then there is a 5% chance that the asset will lose \$300 or more in the next month. We would say that the 5% one-month VaR for this asset is \$300. A different asset might have a 5% one-month VaR of \$200. In that case, the latter asset would have a 5% chance of a loss of \$200 or more in the next month. Larger values of VaR indicate that the asset has greater risk, because larger values indicate that, at the specified confidence level (usually 99% or 95%), the asset has a more extreme threshold of losses.

In *Portfolio construction*, we built four example portfolios: an equally weighted portfolio, a minimum variance portfolio, a minimum variance portfolio with only positive weights, and a portfolio that maximized the return-to-risk ratio. We now assess these portfolios based on their VaR by using the `finvalrisk` command.

```
. finvalrisk p*, percentiles(1)
Value-at-risk percentiles
Number of obs = 779
Sample: 1955m2 thru 2019m12
```

	Empirical	Normal
1%		
Equal	-8.405127	-5.723881
MinVar	-1.831169	-1.728795
NoShort	-2.272815	-2.187996
MaxSharpe	-.7692161	-.787994

The output shows percentiles for each portfolio corresponding to the 1% VaR or, equivalently, VaR at the 99% confidence level. The interpretation is that there is a 1% chance that the portfolio returns this amount or less in the next month. The percentiles reported here are negative returns, but the VaR is typically written as positive, representing the potential loss. The default output shows two methods of calculating VaR. The first method is the empirical method, which takes the appropriate quantile (in this case, the 1% quantile) of the empirical distribution of the data. We see that the equally weighted portfolio has a 1% VaR of 8.41%, so it has a 1% chance of having a monthly return of -8.41% or worse. Meanwhile, the minimum variance portfolio has a 1% chance of having a monthly return -1.83% or worse. The minimum variance portfolio without short selling, which is restricted to having positive weights, suffers somewhat from this restriction; its 1% VaR is 2.27%. Finally, the maximum Sharpe ratio portfolio has a 1% VaR of only 0.77%.

The second method uses a normal approximation fit to the returns. This approach models returns as being drawn independently from a normal distribution, where the mean and standard deviation of the normal distribution are equal to the mean and standard deviation of the observed return series. The VaRs estimated from the normal approximation here are mostly similar to the empirical VaRs, except for the equally weighted portfolio. Normal-based VaR for the equally weighted portfolio is 5.72%, whereas its empirical VaR is 8.41%, indicating that the equally weighted portfolio has especially fat tails and that the normal approximation may be underestimating VaR for this portfolio.

Different quantiles can be requested. Model-based VaR can also be computed, in which a time-series model is fit to the asset returns and the model-implied VaR is reported. Available models include the autoregressive moving-average model (ARMA), the exponentially weighted moving-average model, and the family of autoregressive conditional heteroskedasticity models. See [FIN] **finvalrisk** for all the features of this command.

Other time-series tools

In this section, we demonstrate the use of some of the broader time-series tools of Stata that are applicable to financial data. This is not an exhaustive compendium. See [TS] **Time series** for an overview of Stata's time-series capabilities. While all time-series commands may be useful for analyzing financial data, we focus here on some that are frequently used.

Modeling returns

We would like to investigate some of the time-series properties of the log market return. A longstanding question in financial research has been the degree of predictability of asset returns. The pure random walk model, for instance, proposes that returns cannot be forecasted from prior returns. We explore predictability here by fitting some simple ARMA models to the log market return.

The `arimasoc` command fits ARMA models with various combinations of autoregressive and moving-average components to find a specification that describes the dynamics well. To explore specifications for the log market returns with at most 6 autoregressive lags and at most 1 moving-average lag, we type

```
. arimasoc lnr_mkt, maxar(6) maxma(1)
Fitting models (14): .............. done
Lag-order selection criteria
Sample: 1955m2 thru 2019m12                     Number of obs = 779
```

Model	LL	df	AIC	BIC	HQIC
ARMA(0,0)	-2079.026	2	4162.052	4171.368	4165.635
ARMA(0,1)	-2054.852	3	4115.703	4129.677	4121.078
ARMA(1,0)	-2056.289	3	4118.578	4132.552	4123.953
ARMA(1,1)	-2054.838	4	4117.677	4136.309	4124.843
ARMA(2,0)	-2055.202	4	4118.405	4137.037	4125.571
ARMA(2,1)	-2053.431	5	4116.863	4140.153	4125.821
ARMA(3,0)	-2054.478	5	4118.956	4142.246	4127.915
ARMA(3,1)	-2053.852	6	4119.705	4147.653	4130.455
ARMA(4,0)	-2053.801	6	4119.603	4147.551	4130.353
ARMA(4,1)	-2053.459	7	4120.917	4153.523	4133.459
ARMA(5,0)	-2051.501	7	4117.002	4149.608	4129.543
ARMA(5,1)	-2048.608	8	4113.215	4150.479	4127.548
ARMA(6,0)	-2044.992	8	4105.983	4143.247	4120.316
ARMA(6,1)	-2044.971	9	4107.942	4149.864	4124.067

```
Selected (max) LL:   ARMA(6,1)
Selected (min) AIC:  ARMA(6,0)
Selected (min) BIC:  ARMA(0,1)
Selected (min) HQIC: ARMA(6,0)
```

A more complex model will better fit the data but brings costs in the form of higher complexity. Model selection criteria weigh model fit against model complexity. Different criteria penalize model complexity to differing degrees, so the selected model can differ by criterion. Here the Akaike information criterion (AIC) and Hannan–Quinn information criterion (HQIC) select an autoregressive model with 6 lags, while the Bayesian information criterion (BIC) selects a model with 1 moving-average lag. We will investigate the autoregressive model with 6 lags. To learn more about ARIMA model selection, see [TS] **arimasoc**.

The arima command fits ARMA models. To fit a model with 6 autoregressive lags, we use the ar() option:

```
. arima lnr_mkt, ar(1/6)
(setting optimization to BHHH)
Iteration 0:  Log likelihood = -2044.9949
Iteration 1:  Log likelihood =  -2044.992
Iteration 2:  Log likelihood = -2044.9917
Iteration 3:  Log likelihood = -2044.9916
Iteration 4:  Log likelihood = -2044.9916
(switching optimization to BFGS)
Iteration 5:  Log likelihood = -2044.9916
Iteration 6:  Log likelihood = -2044.9916
ARIMA regression
Sample: 1955m2 thru 2019m12                     Number of obs     =        779
                                                Wald chi2(6)      =      76.12
Log likelihood = -2044.992                      Prob > chi2       =     0.0000
```

lnr_mkt	Coefficient	OPG std. err.	z	P>|z|	[95% conf.	interval]
lnr_mkt						
_cons	.5761579	.1778133	3.24	0.001	.2276502	.9246656
ARMA						
ar						
L1.	.257773	.0330009	7.81	0.000	.1930925	.3224535
L2.	-.0610409	.0344089	-1.77	0.076	-.1284811	.0063993
L3.	.0428401	.0338751	1.26	0.206	-.0235538	.1092341
L4.	.0129247	.0324971	0.40	0.691	-.0507685	.0766178
L5.	.1095655	.0333176	3.29	0.001	.0442642	.1748667
L6.	-.129245	.029311	-4.41	0.000	-.1866935	-.0717964
/sigma	3.34029	.0611667	54.61	0.000	3.220406	3.460175

```
Note: The test of the variance against zero is one sided, and the two-sided
      confidence interval is truncated at zero.
```

The results indicate that market returns are mildly forecastable. A 1 percentage point higher return in a given period forecasts a 0.26 percentage point higher return in the subsequent period. The coefficients on lags greater than the one period are smaller, ranging from −0.13 to 0.11. We conclude there is mild evidence against a pure random walk.

Finally, we store the ARMA model results for later use.

```
. estimates store armamodel
```

The model selection we performed earlier used in-sample information criteria that trade off model fit and model complexity. But this is not the only approach available. We could have instead chosen a model according to out-of-sample fit, for example, by minimizing out-of-sample mean squared error. We could split the sample, say, by holding out a fraction of the most recent observations and assess fit on the held-out observations. To hold out all observations after the year 1999, we can first create a variable that is 1 for observations before the year 2000 and 0 afterward.

```
. generate insample = (datem < tm(2000m1))
```

Then, we could fit a number of models to the in-sample observations. As a baseline, an AR(0) model has no dependence at all, simply estimating a mean.

```
. arima lnr_mkt if insample
```

The model's predictions could then be obtained with `predict`, and the mean squared error of the prediction could be generated as a new variable.

```
. predict double ar0
. generate double mse0 = (lnr_mkt - ar0)^2
```

Next, an AR(1) model could add some structure and would be fit by typing

```
. arima lnr_mkt if insample, ar(1)
```

This model's predictions could also be obtained with `predict`, and the mean squared error of the prediction could again be generated as a new variable.

```
. predict double ar1
. generate double mse1 = (lnr_mkt - ar1)^2
```

We could then do the same for an AR(2) model,

```
. arima lnr_mkt  if insample, ar(1/2)
. predict double ar2
. generate double mse2 = (lnr_mkt - ar1)^2
```

and continue fitting as many models as theory suggests are plausible. If we did so, say, up to an AR(6) model, then the in-sample mean squared errors can be compared with

```
. summarize mse* if insample
```

Variable	Obs	Mean	Std. dev.	Min	Max
mse0	539	11.5329	23.26755	.0000422	199.0792
mse1	539	10.86623	21.21991	5.23e-09	168.1059
mse2	539	10.83973	21.28822	.0000248	172.7485
mse3	539	10.839	21.28625	1.50e-06	171.6705
mse4	539	10.8258	21.33891	1.78e-06	176.124
mse5	539	10.79703	21.17437	4.96e-06	169.7509
mse6	539	10.68633	20.4926	.0000626	164.5492

The in-sample mean squared error of the baseline model, with no autoregressive structure, is 11.53. Adding structure reduces in-sample mean squared error, but the gains from adding lags beyond the first is small. Still, these are in-sample observations, and more important are out-of-sample tests.

The out-of-sample mean squared errors can be compared with

```
. summarize mse* if !insample
```

Variable	Obs	Mean	Std. dev.	Min	Max
mse0	240	13.67175	41.57669	.0000284	551.7558
mse1	240	12.91045	37.46704	.0019075	487.759
mse2	240	12.86826	37.26065	.000111	482.0199
mse3	240	12.84568	37.09217	.0003288	479.2123
mse4	240	12.81103	36.93194	.0000197	474.2833
mse5	240	12.68043	37.2259	.0000468	480.6316
mse6	240	12.35502	36.11761	.0016538	467.5045

The selected model would be the one with the smallest out-of-sample prediction error. Looking across the models, again we see that there is a reduction in mean squared error by including some autoregressive structure rather than none at all. However, adding structure beyond the first brings only modest reduction in mean squared error. The best model is the most complex, the AR(6), though it is only marginally better than the AR(1). For more information about fitting ARIMA models in Stata, see [TS] **arima**.

This out-of-sample procedure is one option among many and one that focused on lagged values as independent variables. We could instead use `bmaregress` to perform Bayesian model averaging of multiple models. Or apply the `h2oml` and `python` commands to analyze financial returns using the vast machine learning toolkits in H2O and Python.

Modeling volatility

Next we investigate persistence in volatility. Previously, we were interested in understanding the predictability of returns themselves. Now we turn to models of volatility to investigate the question of whether periods of high, or low, volatility are themselves autocorrelated.

As an initial pass, we use `estat archlm`, a tool available after `regress`, to check for the presence of time-varying volatility (specifically, conditional heteroskedasticity) in the errors from an autoregressive model.

```
. regress lnr_mkt L(1/6).lnr_mkt
```

Source	SS	df	MS		
Model	787.378292	6	131.229715	Number of obs	= 773
				F(6, 766)	= 11.67
Residual	8614.83162	766	11.2465165	Prob > F	= 0.0000
				R-squared	= 0.0837
				Adj R-squared	= 0.0766
Total	9402.20991	772	12.1790284	Root MSE	= 3.3536

lnr_mkt	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
lnr_mkt						
L1.	.2581365	.0357805	7.21	0.000	.1878971	.3283758
L2.	-.0631958	.036761	-1.72	0.086	-.1353601	.0089684
L3.	.0428656	.0368258	1.16	0.245	-.029426	.1151571
L4.	.0132665	.0368247	0.36	0.719	-.0590229	.0855559
L5.	.1078361	.0367977	2.93	0.003	.0355998	.1800724
L6.	-.1291023	.0357914	-3.61	0.000	-.1993632	-.0588415
_cons	.4280706	.1267854	3.38	0.001	.1791825	.6769587

```
. estat archlm, lag(1/6)
LM test for autoregressive conditional heteroskedasticity (ARCH)
```

lags(*p*)	chi2	df	Prob > chi2
1	4.771	1	0.0289
2	5.012	2	0.0816
3	7.083	3	0.0693
4	7.651	4	0.1052
5	8.833	5	0.1159
6	24.327	6	0.0005

H0: no ARCH effects *vs.* H1: ARCH(*p*) disturbance

The estat archlm command performs a Lagrange multiplier test for the presence of autoregressive conditional heteroskedasticity (ARCH) behavior in the errors of the regression. We see sufficient evidence of ARCH behavior, so we will investigate it more fully.

Continuing with the suggested specification from the previous section, here we use the arch command to fit an autoregressive model of order 6 to capture any dependence in the level of returns and add GARCH(1,1) terms to capture time-varying volatility.

```
. arch lnr_mkt, ar(1/6) arch(1) garch(1)
(setting optimization to BHHH)
Iteration 0:  Log likelihood = -2037.5614
Iteration 1:  Log likelihood = -2029.6178
Iteration 2:  Log likelihood = -2023.7281
Iteration 3:  Log likelihood = -2021.6537
Iteration 4:  Log likelihood = -2020.3526
(switching optimization to BFGS)
Iteration 5:  Log likelihood = -2019.7859
Iteration 6:  Log likelihood = -2019.1407
Iteration 7:  Log likelihood =  -2019.067
Iteration 8:  Log likelihood = -2019.0626
Iteration 9:  Log likelihood = -2019.0623
Iteration 10: Log likelihood = -2019.0622
Iteration 11: Log likelihood = -2019.0622
ARCH family regression -- AR disturbances
Sample: 1955m2 thru 2019m12                     Number of obs     =        779
                                                Wald chi2(6)      =      35.92
Log likelihood = -2019.062                      Prob > chi2       =     0.0000
```

lnr_mkt	Coefficient	OPG std. err.	z	P>|z|	[95% conf.	interval]
lnr_mkt						
_cons	.7233727	.1403091	5.16	0.000	.448372	.9983734
ARMA						
ar						
L1.	.2211217	.0429604	5.15	0.000	.1369209	.3053225
L2.	-.0539058	.03845	-1.40	0.161	-.1292664	.0214548
L3.	.0299586	.0386123	0.78	0.438	-.0457201	.1056374
L4.	.0025786	.0389335	0.07	0.947	-.0737297	.078887
L5.	.0760806	.0380917	2.00	0.046	.0014223	.1507389
L6.	-.0951334	.0329823	-2.88	0.004	-.1597775	-.0304892
ARCH						
arch						
L1.	.1382138	.0266536	5.19	0.000	.0859737	.1904539
garch						
L1.	.7834001	.0480069	16.32	0.000	.6893084	.8774919
_cons	.9715679	.3603296	2.70	0.007	.2653348	1.677801

The ARCH and GARCH terms are both positive, indicating persistence in volatility. The autoregressive model parameters have changed slightly, but the main features remain: average returns continue to be slightly forecastable. If returns are 1 percentage point higher than average in the current period, we would forecast the return being 0.22 percentage points higher than average in the subsequent period, all else equal. For more information on ARCH and GARCH models, see [TS] **arch**.

Forecasting

Stata has a forecasting environment to make predictions given a model; see [TS] **forecast**. We can, for example, use the previously fit ARMA model to generate a forecast for market returns over the first six months out of sample, along with confidence intervals that account for estimation uncertainty and the random errors arising from the stochastic nature of the model.

To begin the process, we add six periods (months) to the end of the dataset. These will be the months for which we forecast market returns.

```
. tsappend, add(6)
```

To start a forecast, we use `forecast create` and give the forecast model a name.

```
. forecast create arimafc
  Forecast model arimafc started.
```

We add the previously fit model `armamodel` to the forecast using `forecast estimates`.

```
. forecast estimates armamodel
  Added estimation results from arima.
  Forecast model arimafc now contains 1 endogenous variable.
```

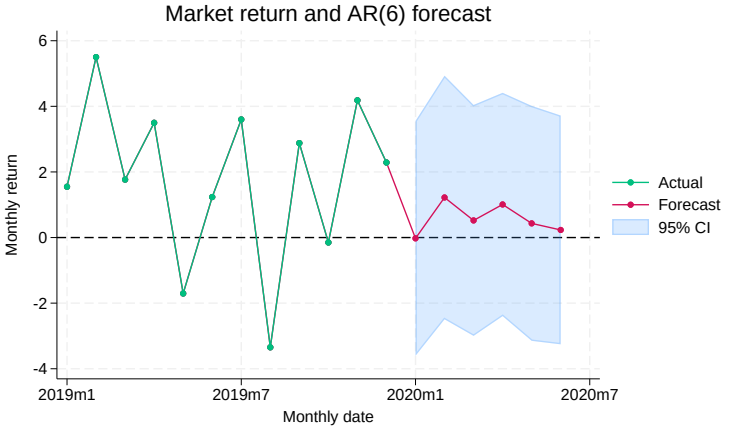
We now solve the forecast model, where "solve" means to actually make our forecast. We use the `prefix()` option to add a prefix to the newly generated forecast variables and use the `begin()` option to begin the forecast at a certain date. In the `armamodel` estimates, the dependent variable is the log market return `lnr_mkt`. Thus, the newly created forecast will be stored in a variable called `arma_lnr_mkt`. We use the first month of 2020 as our beginning forecast date.

```
. forecast solve, prefix(arma_) begin(tm(2020m1))
> simulate(betas errors, statistic(stddev, prefix(sd_)) reps(100))
Computing dynamic forecasts for model arimafc.
------------------------------------------------------------

Starting period:  2020m1
Ending period:    2020m6
Forecast prefix:  arma_

2020m1:  ..............
2020m2:  ..............
2020m3:  ..............
2020m4:  ..............
2020m5:  ..............
2020m6:  ..............

Performing simulations (100): ..................................................... 50
.....................................................  100

Forecast 1 variable spanning 6 periods.
------------------------------------------------------------
```

The second line of the `forecast solve` command above specifies how uncertainty is treated. We create an additional variable with prefix `sd_` to store the standard deviation of the forecast variable. This standard deviation incorporates two sources of uncertainty. The specification `betas` accounts for the fact that the parameters were estimated and thus subject to uncertainty. The specification `errors` accounts for the randomness inherent in the stochastic process. For more details on uncertainty specification, see example 2 of [TS] **forecast solve**.

The output from `forecast solve` informs us that we made predictions for 6 forecast periods and that our uncertainty estimates were drawn from 100 simulations of the parameter estimates (`betas`) and stochastic uncertainty (`errors`).

A graph of the forecast shows the following:



Producing this graph takes two steps. First, we produce variables for the upper and lower confidence band. Second, we graph the actual market return, the forecasted return, and the confidence band. We use some of the tools in the `graph twoway` command to add nice titles to the axes, add nice labels to the series, and order the legend.

```
. generate lnr_mkt_up = arma_lnr_mkt + invnormal(0.975) * sd_lnr_mkt
(780 missing values generated)
. generate lnr_mkt_dn = arma_lnr_mkt + invnormal(0.025) * sd_lnr_mkt
(780 missing values generated)
. twoway  (rarea lnr_mkt_dn lnr_mkt_up datem if tin(2019m1,), color(stblue%20))
>         (connected arma_lnr_mkt datem if tin(2019m1,))
>         (connected lnr_mkt datem if tin(2019m1, )),
> legend(label(1 "95% CI")) legend(label(2 "Forecast"))
> legend(label(3 "Actual")) legend(order( 3 2 1))
> ytitle("Monthly return") title("Market return and AR(6) forecast") yline(0)
```

Turning to the results, we see that the forecast eventually settles at the constant reported in the `arima` output, which is close to the sample average market return. The confidence intervals are wide, reflecting considerable uncertainty in forecasted returns.

The `forecast` command allows for dynamic, multistep forecasting after virtually all Stata estimation commands. Also of interest in a time-series context are the smoothers, which can also be used for forecasting. See [TS] **tssmooth** for details.

Multivariate models

So far, we have mainly examined one time series in isolation, the log market return. But it is also possible to analyze time series jointly. The main benefit of doing so arises when the two series exhibit jointly dependent behavior, either contemporaneously or over time. As an example, we can jointly model the market return and the risk-free rate of return, `lnr_mkt` and `rf`, in a vector autoregression model. Whereas a univariate autoregression allows a time series to depend on its own lags, a vector autoregression allows a time series to depend on both its own lags and on lags of other series. The following uses the `var` command to fit a vector autoregression model for the two series, using two lags of each variable as explanatory variables.

```
. var lnr_mkt rf
Vector autoregression
Sample: 1955m4 thru 2019m12                     Number of obs     =       777
Log likelihood =  -516.1938                     AIC               =  1.354424
FPE            =   .0132822                     HQIC              =  1.377473
Det(Sigma_ml)  =   .0129447                     SBIC              =   1.41434
Equation           Parms      RMSE     R-sq      chi2     P>chi2
------------------------------------------------------------------------------
lnr_mkt                5   3.37616   0.0717   59.99281   0.0000
rf                     5   .033964   0.9855   52702.85   0.0000
------------------------------------------------------------------------------
------------------------------------------------------------------------------
             |      Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
lnr_mkt      |
     lnr_mkt |
         L1. |      .2397095   .0357808     6.70   0.000     .1695804    .3098385
         L2. |     -.0447416   .0358032    -1.25   0.211    -.1149146    .0254314
             |
          rf |
         L1. |     -9.924366   3.270127    -3.03   0.002    -16.3337   -3.515035
         L2. |      9.438268   3.269906     2.89   0.004     3.02937    15.84717
             |
       _cons |      .6519097   .2094337     3.11   0.002     .2414272    1.062392
-------------+----------------------------------------------------------------
rf           |
     lnr_mkt |
         L1. |      .0008055    .00036     2.24   0.025        .0001    .001511
         L2. |      .0009662   .0003602     2.68   0.007     .0002602    .0016721
             |
          rf |
         L1. |      1.378865   .0328971    41.91   0.000     1.314388    1.443342
         L2. |     -.3897623   .0328949   -11.85   0.000    -.4542351   -.3252894
             |
       _cons |      .0032188   .0021069     1.53   0.127    -.0009106    .0073482
------------------------------------------------------------------------------
```

In the first equation, we are regressing `lnr_mkt` on its first two lags and on the first two lags of the risk-free rate. We see that market returns are mildly forecastable in that the lagged market return provides information on the current market return. But there is also cross-dependence: the market return also reacts to the lagged risk-free rate. The first cross-lag coefficient is -9.92, indicating that an increase in the risk-free rate leads to negative market returns one period (one month) later. The second lag is actually positive, of about the same magnitude. This too can be interpreted: it means that the market return reacts to the change in the risk-free rate, not the level. To see this, write the estimated equation in the following way. Letting r_t^m be the market return and r_t^f be the risk-free rate, the first equation is approximately

$$r_t^m = 0.65 + 0.24 \times r_{t-1}^m - 0.04 \times r_{t-2}^m - 9.9 \times r_{t-1}^f + 9.4 \times r_{t-2}^f$$

where all the numbers come from the coefficients in the first equation in the output table. A little algebraic rearranging gives

$$r_t^m = 0.65 + 0.24 \times r_{t-1}^m - 0.04 \times r_{t-2}^m - 0.5 \times r_{t-1}^f - 9.4 \times (r_{t-1}^f - r_{t-2}^f)$$

which is more interpretable: the market return responds to the lagged level of the risk-free rate and reacts to lagged changes in the risk-free rate.

Turning to the risk-free rate itself, we see that it responds to its own lags and also responds in a small way to the lagged market return.

The model is illustrative. We could have chosen different lag lengths, perhaps using `varsoc` to choose the lag order similarly to how we used `arimasoc` earlier. In addition, after fitting a vector autoregression, we can obtain unconditional forecasts with `fcast compute` or `fcast graph`, assess the impact of shocks with `irf`, and test for the significance of those cross-lag terms jointly with `vargranger`. For much more information on how to fit, analyze, and interpret vector autoregressions, see [TS] **var**.

The `var` analogue for the study of patterns of conditional volatility in a multivariate context is a multivariate GARCH model, implemented in the `mgarch` command. In such a model, periods of volatility cluster together, just as in a univariate GARCH model; but unlike a univariate GARCH model, the multivariate GARCH model can also characterize correlations in volatility across time series. We can ask, for instance, whether one series (like the market return) is volatile precisely when another series (like the risk-free rate) is volatile or whether the two experience periods of volatility independently. This kind of question is often asked of multiple asset returns or exchange rates across countries, in which the patterns of correlated volatility can indicate the presence of linkages in the financial markets across countries. For more on the multivariate GARCH model, including four different submodels of volatility and the ways to interpret them, see [TS] **mgarch**.

Date and time management

At the beginning of this introduction, we used `tsset` to declare to Stata that we are working with time-series data. This section provides additional information on `tsset`, especially for use with daily data. Much financial data arrive daily but with regular and expected gaps. Financial markets are open only on weekdays, so that the "lag" of a Monday is the previous Friday, not the preceding Sunday. Financial markets can be closed for holidays, for example, Christmas and Thanksgiving (in the United States) or Showa Day (in Japanese markets, near the end of April) or Chinese New Year (in Chinese markets). These regularly observed holidays cause gaps in the data that are only apparent, not true gaps.

Stata has the concept of business calendars to handle such situations. A business calendar begins with daily data, then applies some rules specified in the calendar to appropriately treat certain gaps. We provide a few example business calendars for some stock exchanges around the world and encourage you to make your own calendar to fit your specific needs.

A business calendar is a file, ending in `.stbcal`, containing rules for how certain gaps should be handled. Let's look at one example calendar for the New York Stock Exchange for the years 2024–2028.

begin _nyse.stbcal

```
*! version 1.0.0 05may2026
*  New York Stock Exchange business calendar (_nyse.stbcal)
*  Source: https://www.nyse.com/markets/hours-calendars
*  Range: January 1, 2024, to December 31, 2028
version 19.5
purpose "NYSE business calendar"
dateformat dmy
range 02jan2024 29dec2028
centerdate 02jan2024

* Weekends
omit dayofweek(Sa Su)

* New Year's Day
omit date 01jan2024
omit date 01jan2025
omit date 01jan2026
omit date 01jan2027

* Martin Luther King Jr. Day
omit date 15jan2024
omit date 20jan2025
omit date 19jan2026
omit date 18jan2027
omit date 17jan2028

* Washington's Birthday
omit date 19feb2024
omit date 17feb2025
omit date 16feb2026
omit date 15feb2027
omit date 21feb2028

* Good Friday
omit date 19mar2024
omit date 18apr2025
omit date 03apr2026
omit date 26mar2027
omit date 14apr2028

* Memorial Day
omit date 27may2024
omit date 26may2025
omit date 25may2026
omit date 31may2027
omit date 29may2028

* Juneteenth National Independence Day
omit date 19jun2024
omit date 19jun2025
omit date 19jun2026
omit date 18jun2027
omit date 19jun2028
```

```
* Independence Day
omit date 04jul2024
omit date 04jul2025
omit date 03jul2026
omit date 05jul2027
omit date 04jul2028

* Labor Day
omit date 02sep2024
omit date 01sep2025
omit date 07sep2026
omit date 06sep2027
omit date 04sep2028

* Thanksgiving Day
omit date 28nov2024
omit date 27nov2025
omit date 26nov2026
omit date 25nov2027
omit date 23nov2028

* Christmas Day
omit date 25dec2024
omit date 25dec2025
omit date 25dec2026
omit date 24dec2027
omit date 25dec2028
```

end _nyse.stbcal

The business calendar begins with some comments describing its date of construction, source material, and date range. The first few lines fix the `version`, specify a `purpose`, identify the date format, give the range of the calendar, and finally provide a `centerdate`, the date that, for the calendar, represents "time 0". Days are counted in positive integers after the center date and as negative integers before the center date.

Next, the `omit dayofweek(Sa Su)` command marks all weekends as anticipated gaps. Then, New Year's Day is omitted (the first day in January, each year). And we see Martin Luther King Jr. Day omitted; its date changes slightly each year because it is the third Monday in January. Example calendars are provided for five stock exchanges, showing the basic structure of a business calendar and how to carefully omit dates. To learn how to download these example calendars, see [D] **Datetime business calendars** and [D] **bcal**.

As a second example, let's look at _stock.stbcal, which is designed to work with stocks.dta.

begin _stock.stbcal

```
* version 1.0.0  07may2026
* Business calendar for -webuse stocks-
* Range: January 1, 2003, to December 31, 2010
version 19.5
purpose "Stock calendar for 2003-2010"
dateformat dmy
range 02jan2003 31dec2010
centerdate 02jan2003

* Weekends
omit dayofweek (Sa Su)

* New Year's Day
omit date 01jan*
omit date 01jan* and (+1) if dow(Su)
omit date 02jan2007

* Martin Luther King Jr. Day
omit dowinmonth +3 Mo of Jan

* President's Day
omit dowinmonth +3 Mo of Feb
  (output omitted)

* Memorial Day
omit dowinmonth -1 Mo of May
  (output omitted)

* Christmas
omit date 25dec*
omit date 25dec* and (-1) if dow(Sa)
omit date 25dec* and (+1) if dow(Su)
```

end _stock.stbcal

In this calendar, we again set version, specify purpose, set dateformat, and give range over which the calendar is valid. The calendar is valid for dates between January 2, 2003, and December 31, 2010, with a center date (0 date) of January 2, 2003. Here are some comments on the syntax and rules:

- As before, we omit all weekends via omit dayofweek(Sa Su).
- For New Year's Day, we use the wildcard * to mark all January 1s as an anticipated gap, regardless of year. Also, if January 1 falls on a Sunday, we additionally omit the following Monday.
- For Martin Luther King Jr. Day, we use some of the rules available in business calendars. Specifically, we omit the third Monday in each January with omit dowinmonth +3 Mo of Jan; this one line marks the third Monday in each January as an anticipated gap.
- We do the same for President's Day, omitting the third Monday in each February.
- For Memorial Day, which falls on the final Monday of each May, we use the rule -1 Mo of May, which omits this date.

Many other rules can be specified. See [D] **Datetime business calendars** for details.

Once you have carefully constructed the correct calendar for your application, how do you apply it to the data? Let's look at `stocks.dta`.

```
. use https://www.stata-press.com/data/r19/stocks, clear
(Data from Yahoo! Finance)
. describe
Contains data from https://www.stata-press.com/data/r19/stocks.dta
 Observations:         2,015                  Data from Yahoo! Finance
    Variables:             5                  9 May 2024 11:48
-------------------------------------------------------------------------------
Variable      Storage   Display    Value
    name         type    format    label      Variable label
-------------------------------------------------------------------------------
date            int     %td                   Date
t               int     %9.0g                 Generic time variable
toyota          float   %9.0g                 Daily return on Toyota stock
nissan          float   %9.0g                 Daily return on Nissan stock
honda           float   %9.0g                 Daily return on Honda stock
-------------------------------------------------------------------------------
Sorted by: t
. tsset date
Time variable: date, 02jan2003 to 31dec2010, but with gaps
        Delta: 1 day
. tsreport
Time variable: date
-----------------------------
Starting period = 02jan2003
Ending period   = 31dec2010
Number of obs   =     2,015
Number of gaps  =       433
```

We used `tsreport` to get an overview of the span of the data and to alert us to the presence of gaps. This dataset has daily data from January 2, 2003, to December 31, 2010. If we list the first few observations of the Toyota stock price and its lag, we obtain the following:

```
. list date toyota L.toyota in 1/10
```

	date	toyota	L. toyota
1.	02jan2003	.0151675	.
2.	03jan2003	.0048201	.0151675
3.	06jan2003	.0199587	.
4.	07jan2003	-.0133226	.0199587
5.	08jan2003	-.0270011	-.0133226
6.	09jan2003	.0116346	-.0270011
7.	10jan2003	-.0093722	.0116346
8.	13jan2003	.0016935	.
9.	14jan2003	-.0062234	.0016935
10.	15jan2003	-.0039806	-.0062234

There are unintended gaps due to weekends and holidays. So let's load our stock calendar and apply it.

```
. bcal webcopy stock
(business calendar _stock.stbcal copied to working directory)
. generate dateb = bofd("_stock", date)
```

We used the generate command to create a new variable, dateb. This new variable starts from the daily date date and applies the rules in stocks.stbcal to create a running date that omits the gaps we specified.

```
. tsset dateb
Time variable: dateb, 0 to 2014
        Delta: 1 unit
. format dateb %tb_stock
. tsreport
Time variable: dateb
-----------------------------
Starting period = 02jan2003
Ending period   = 31dec2010
Number of obs   =     2,015
Number of gaps  =         0
```

The business calendar has rid us of the gaps because it correctly recognizes holidays and weekends. If we look at the current and lagged values of the first few observations on the Toyota stock, we see

```
. list date dateb toyota L.toyota in 1/10
```

	date	dateb	toyota	L. toyota
1.	02jan2003	02jan2003	.0151675	.
2.	03jan2003	03jan2003	.0048201	.0151675
3.	06jan2003	06jan2003	.0199587	.0048201
4.	07jan2003	07jan2003	-.0133226	.0199587
5.	08jan2003	08jan2003	-.0270011	-.0133226
6.	09jan2003	09jan2003	.0116346	-.0270011
7.	10jan2003	10jan2003	-.0093722	.0116346
8.	13jan2003	13jan2003	.0016935	-.0093722
9.	14jan2003	14jan2003	-.0062234	.0016935
10.	15jan2003	15jan2003	-.0039806	-.0062234

This result is as desired.

Summing up

This entry has introduced the fin suite of commands for processing asset prices, creating returns, creating portfolios, summarizing returns, running financial regressions, and assessing VaR. It has also provided a brief introduction to some of Stata's time-series commands that are relevant for finance, like arima, arch, and var. This introduction was not comprehensive; we could have shown you mgarch for multivariate volatility models, vec for models of variables with common trends, pca and factor for static unobserved factor models, or dfactor for unobserved dynamic factor models.

Also see

[FIN] **Intro** — Introduction to financial statistics+

finportfolio — Financial portfolio selection[+]

[+]This command is part of StataNow.

Description

`finportfolio` calculates portfolio weights given a collection of returns and an optimization objective. Equal weights, minimum variance weights, and maximum Sharpe ratio weights are supported. Weights can be calculated with and without short selling (negative weights).

Quick start

Compute global minimum variance portfolio weights from return variables `r1`, `r2`, `r3`, and `r4`

```
finportfolio minvariance r1 r2 r3 r4
```

Same as above, but require nonnegative weights (no short selling)

```
finportfolio minvariance r1 r2 r3 r4, noshort
```

Find the minimum variance portfolio that hits a target return of 0.05

```
finportfolio minvariance r1 r2 r3 r4, target(0.05)
```

Same as above, but require nonnegative weights

```
finportfolio minvariance r1 r2 r3 r4, target(0.05) noshort
```

Find the global minimum variance portfolio, and sort the assets by portfolio weight in the output

```
finportfolio minvariance r1 r2 r3 r4, sort(ascending)
```

Compute the portfolio return using equal weights for returns `r1` through `r10`

```
finportfolio equal r1-r10
```

Set custom portfolio weights for three return variables

```
matrix myweights = (0.25, 0.5, 0.25)
matrix colnames myweights = r1 r2 r3
finportfolio fixed r1-r3, at(myweights)
```

Find the portfolio weights that maximize the Sharpe (return-to-risk) ratio with risk-free rate `rf`

```
finportfolio maxsharpe r1-r10, rfrate(rf)
```

Menu

Statistics > Financial statistics > Financial portfolio selection

Syntax

```
finportfolio method varlist [if] [in] [, options]
```

method	Description
minvariance	minimum variance weights
maxsharpe	maximum Sharpe ratio weights
equal	equal weights
fixed	fixed weights

options	Description
Main	
rfrate(*varname*)	specify variable containing the risk-free rate
generate(*varname*[, label("*label*") replace])	generate variable containing portfolio returns
novarlabel	display variable names rather than variable labels
sort(*sortmethod*)	specify sort order for assets in the output
minvar_options	options for method minvariance
maxsharpe_options	options for method maxsharpe
fixed_option	option for method fixed

You must tsset your data before using finportfolio; see [TS] **tsset**.

rfrate() may contain time-series operators; see **[U] 11.4.4 Time-series varlists**.

collect is allowed; see **[U] 11.1.10 Prefix commands**.

sortmethod	Description
ascending	sort assets by ascending values of weights
descending	sort assets by descending values of weights
alphabetical	sort by asset label

minvar_options	Description
Main	
target(#)	specify target return
noshort	disallow short selling (negative weights)
gamma(#)	tuning parameter when noshort is specified
Maximization	
maximize_options	control the optimization when noshort is specified

maxsharpe_options	Description
Main	
noshort	disallow short selling (negative weights)
gamma(#)	tuning parameter when noshort is specified
Maximization	
maximize_options	control the optimization when noshort is specified

fixed_option	Description
Main	
* `at(`*wspec*`)`	use specified weights

*`at()` is required.

Options

Options are presented under the following headings:

Options for all methods

Main

`rfrate(`*varname*`)` specifies a risk-free asset. When `rfrate()` is specified, the Sharpe ratio is $(r-r^f)/\sigma$, where r is the portfolio return, r^f is the average risk-free rate, and σ is the portfolio standard deviation. When `rfrate()` is not specified, the risk-free rate is set to 0.

`generate(`*varname*`[ , label("`*label*`") replace ])` generates a new variable to store the return series of the portfolio that `finportfolio` creates. The generated variable is stored in `double` precision, regardless of the default storage type set by using `set type`.

`label("`*label*`")` specifies the variable label for the newly generated variable.

`replace` allows an existing variable named *varname* to be replaced.

`novarlabel` specifies that the variable names be displayed instead of the variable labels. By default, variable labels are used to label assets in the output.

`sort(`*sortmethod*`)` specifies how to sort the assets in the portfolio output. By default, assets are listed in the order specified in *varlist*.

`sort(ascending)` sorts assets by their portfolio weight, from lowest weighted to highest weighted.

`sort(descending)` sorts assets by their portfolio weight, from highest weighted to lowest weighted.

`sort(alphabetical)` sorts assets alphabetically by their variable labels or, if the `novarlabel` option is specified, by their variable names.

Options for finportfolio minvariance

Main

`target(#)` specifies the desired rate of return in the units of the assets in *varlist* of `finportfolio`.

`noshort` specifies that all weights must be positive. This is also known as a no-short-selling condition. By default, negative weights are allowed. `noshort` may be used with or without a target return.

`gamma(#)` specifies the penalty term of the barrier function used to compute no-short weights. The smaller `gamma` is, the more that small, nonzero weights are driven toward zero. The penalty term must be positive.

When a target return is specified, the default is $k \times r^{\text{target}}$ with $k = 0.0001$. When no target return is specified, the default is $k \times (r^{\max} + r^{\min})/2$, where $r^{\max}$ is the maximum average return of the returns in *varlist* and $r^{\min}$ is the minimum average return in *varlist*.

Maximization

maximize_options: difficult, technique(nr), technique(dfp), technique(bfgs), iterate(#), [no]log, trace, gradient, showstep, hessian, tolerance(#), ltolerance(#), nrtolerance(#), nonrtolerance, and from(*init_specs*); see [R] **Maximize**.

These options are allowed only with the noshort option.

Options for finportfolio maxsharpe

Main

noshort specifies that all weights must be positive. This is also known as a no-short-selling condition. By default, negative weights are allowed. noshort may be used with or without a target return.

gamma(#) specifies the penalty term of the barrier function used to compute no-short weights. The larger gamma is, the more that small, nonzero weights are driven towards zero. The penalty term must be positive.

The default is $k \times (r^{\max} + r^{\min})/2$, where $k = 0.0001$, $r^{\max}$ is the maximum average return of the returns in *varlist* and $r^{\min}$ is the minimum average return in *varlist*.

Maximization

maximize_options: difficult, technique(nr), technique(dfp), technique(bfgs), iterate(#), [no]log, trace, gradient, showstep, hessian, tolerance(#), ltolerance(#), nrtolerance(#), nonrtolerance, and from(*init_specs*); see [R] **Maximize**.

These options are allowed only with the noshort option.

Option for finportfolio fixed

Main

at(*wspec*) specifies the fixed weights to be used when method fixed is specified. This option is required with method fixed, and the weights must sum to one.

wspec may be one of

matname[, copy]

#[#...]

name = #[*name* = #[...]]

You can specify the fixed weights in one of three ways.

1. You can specify the name of a Stata row vector containing the weights such as at(myrowvec). If your row vector has column names matching the variables in *varlist*, then at() matches the weights accordingly, regardless of the order. You may omit weights from the row vector myrowvec, and if you do so, the weights corresponding to the omitted variables will be zero.

If you specify the copy suboption, for example, at(myrowvec, copy), then the values in myrowvec are used as the weights in a direct positional copy. The first entry is the first weight, the second entry is the second weight, and so forth. When copy is specified, the length of the row vector myrowvec must be equal to the number of variables in *varlist*.

2. You can specify a list of numbers directly, for example, at(.1 .2 .3 .4). The weights in the list are assigned positionally. If the list has fewer elements than the *varlist* of returns, then the remaining weights are set to zero.

3. You can specify a list of variable names and numbers, for example, at(r1=0.7 r3=0.3). The specified variables are given the weight assigned. Weights on other returns, if any, are set to zero.

   The following methods are equivalent for specifying weights of 0.1, 0.3, and 0.6 to returns r1, r2, and r3, respectively:

```
. finportfolio fixed r1 r2 r3, at(0.1 0.3 0.6)

. finportfolio fixed r1 r2 r3, at(r1=0.1 r2=0.3 r3=0.6)

. matrix wgt_default = (0.1, 0.3, 0.6)
. finportfolio fixed r1 r2 r3, at(wgt_default, copy)

. matrix wgt_proper = (0.1, 0.3, 0.6)
. matrix colnames wgt_proper = r1 r2 r3
. finportfolio fixed r1 r2 r3, at(wgt_proper)
```

   If we wish to assign weights only to r1 and r2, we can type any of

```
. finportfolio fixed r1 r2 r3, at(0.1 0.9)

. finportfolio fixed r1 r2 r3, at(r1=0.1 r2=0.9)

. matrix wgt_r1r2 = (0.1, 0.9)
. matrix colnames wgt_r1r2 = r1 r2
. finportfolio fixed r1 r2 r3, at(wgt_r1r2)
```

   If we wish to assign weights only to r1 and r3, we can type any of

```
. finportfolio fixed r1 r2 r3, at(r1=0.1 r3=0.9)

. matrix wgt_r1r3 = (0.1, 0.9)
. matrix colnames wgt_r1r3 = r1 r3
. finportfolio fixed r1 r2 r3, at(wgt_r1r3)
```

Remarks and examples

Remarks are presented under the following headings:

Introduction

Portfolios are combinations of assets. A portfolio is represented by a vector, each entry of which is the fraction of the portfolio that is invested in a particular asset. Portfolios allow an investor to balance the risks and returns of various individual assets.

The `finportfolio` command creates portfolio weights according to various criteria. The simplest portfolio, equal weights, is supported. User-defined weights for tailored portfolios are also supported. The financial literature has introduced several types of portfolios that maximize or minimize some objective. `finportfolio` can find the minimum variance portfolio of a collection of assets, globally or given a target return, with or without short sales. `finportfolio` can also compute portfolio weights that maximize the Sharpe (return-to-risk) ratio, with or without short sales.

For an introduction to portfolio selection, see Hurn et al. (2020, chap. 1). A more detailed discussion of portfolio weights, including minimum variance and maximum Sharpe ratio weights, is in Campbell (2018, chap. 2).

Minimum variance portfolios

The minimum variance method aims to produce the portfolio of assets with the lowest variance. This portfolio can be unconstrained, that is, having global minimum variance, or it can be constructed subject to a specified target return. In the latter case, the portfolio weights calculated are those that reach the target return at the lowest possible variance. In addition, portfolios can be constructed with or without short selling, that is, with or without negative weights.

▷ Example 1: Global minimum variance portfolio

We have stock prices for 25 fictional companies, along with true data on the S&P 500 index and the federal funds rate. We describe some of the variables we will use in our examples.

```
. use https://www.stata-press.com/data/r19/finex
(Fictional stock price data)

. describe datestr datem sp500 fedfunds aaa-cph
```

Variable name	Storage type	Display format	Value label	Variable label
datestr	str11	%11s		String date
datem	int	%tm		Monthly date
sp500	double	%10.0g		S&P 500
fedfunds	float	%9.0g		Federal funds rate
aaa	float	%9.0g		Apex Adventure Airlines
afh	float	%9.0g		Apex Film Holdings
ard	float	%9.0g		Arbor Renewable Dynamics
cph	float	%9.0g		Coastal Property Holdings

The datem variable has been previously declared as the time variable for this time-series dataset by using the tsset command. We confirm this with tsreport.

```
. tsreport
Time variable: datem
-----------------------------
Starting period =  1955m1
Ending period   = 2019m12
Number of obs   =     780
Number of gaps  =       0
```

The data run from the first month of 1955 to the last month of 2019 without any gaps.

Our dataset contains information on stock prices, but portfolios are constructed on the basis of returns. We use finreturns to generate 25 returns series.

```
. quietly finreturns acme-tks, simple(r_) multiply(100)
```

The simple() option requests simple returns, and the argument r_ specifies that the simple returns be stored in a collection of variables called r_acme through r_tks. The multiply(100) option multiplies all simple returns by 100, so that a simple return of, say, 0.005 becomes 0.5, interpretable as a percentage change. The data are monthly, so the generated variables are monthly percentage returns.

Now we are ready to generate some portfolios using the individual stock returns. Let's use returns from stocks aaa, afh, and ard, stored in respective variables r_aaa, r_afh, and r_ard, to obtain a minimum variance portfolio. We use *varlist* to refer to these three variables.

```
. finportfolio minvariance r_aaa-r_ard
Global minimum variance portfolio
Short selling allowed
Number of obs = 779
Sample: 1955m2 thru 2019m12
------------------------------
             |      Weight
-------------+----------------
         AAA |   .0123108
         AFH |   .3763484
         ARD |   .6113408
------------------------------
Portfolio return    = 0.4277
Portfolio std. dev. = 1.4432
Risk-free rate      = 0.0000
Sharpe ratio        = 0.2964
```

The table reports the minimum variance weights for the stock returns. The minimum variance portfolio puts 61% into ARD, 38% into AFH, and the rest into AAA. This portfolio has a historical return of 0.43% at a monthly rate. The corresponding portfolio standard deviation is 1.44%. Because we did not specify a risk-free rate, the risk-free rate defaults to 0. Finally, the Sharpe (return-to-risk) ratio is about 0.30.

◁

Example 2: The minimum variance portfolio with a target return

Continuing with example 1, let's look at the performance of the individual stocks in this portfolio.

```
. summarize r_aaa-r_ard
    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
       r_aaa |        779    .5583062     2.04691  -10.97514   7.364605
       r_afh |        779    .6601578    2.400168  -10.59864   8.662755
       r_ard |        779     .282029    1.871475  -5.362448   6.507083
```

The three stocks have quite different average monthly rates of return: 0.56%, 0.66%, and 0.28%, respectively. The global minimum variance portfolio puts the majority of its weight, 61%, into the stock with the lowest return; this is because that stock also has the lowest variance, and the method is designed to minimize variance. We may be willing to tolerate additional variance above the global minimum if that variance is associated with higher returns. The `target()` option finds the minimum variance portfolio that also hits a specified target return.

The global minimum variance portfolio had an average return of 0.43%. We request a slightly more aggressive portfolio, seeking a 0.5% return monthly. This monthly return corresponds to a 6.16% annual return (annualized using $1.005^{12} - 1$).

```
. finportfolio minvariance r_aaa-r_ard, target(0.5)
Minimum variance portfolio with fixed return
Short selling allowed
Number of obs = 779
Sample: 1955m2 thru 2019m12
------------------------
                  Weight
------------------------
         AAA    .1620223
         AFH     .458066
         ARD    .3799117
------------------------
Portfolio return    = 0.5000 (fixed return)
Portfolio std. dev. = 1.5451
Risk-free rate      = 0.0000
Sharpe ratio        = 0.3236
```

The weight on ARD has fallen from 61% to 38%, and the weights on the other two assets have increased, because the portfolio has shifted away from low-return assets to high-return assets to meet the target return.

Below the weights, we can see the portfolio return is exactly 0.50% per month, as requested. The portfolio standard deviation is now 1.55%, an increase from the global minimum of 1.44% we obtained in example 1. This is a general feature: portfolios trade off risk and return, so higher return is associated with higher risk (variance). The Sharpe ratio, which is return divided by standard deviation, has increased, indicating that this new portfolio has increased returns more than it increased risk.

◁

▷ Example 3: The minimum variance portfolio without short selling

Suppose additional stocks such as CPH were added to the portfolio.

```
. finportfolio minvariance r_aaa-r_cph
Global minimum variance portfolio
Short selling allowed
Number of obs = 779
Sample: 1955m2 thru 2019m12
─────────────┬────────────
             │     Weight
─────────────┼────────────
         AAA │   .0351662
         AFH │   .3867707
         ARD │   .6259844
         CPH │  -.0479213
─────────────┴────────────
Portfolio return    = 0.4312
Portfolio std. dev. = 1.4419
Risk-free rate      = 0.0000
Sharpe ratio        = 0.2991
```

The fourth stock, CPH, has entered with a negative weight. Negative weights are allowed in the unconstrained minimum variance problem and indicate assets that should be sold short. However, not all portfolios allow short selling. The `noshort` option enforces positive weights on all assets in the portfolio, effectively disallowing short selling.

```
. finportfolio minvariance r_aaa-r_cph, noshort
Iteration 0:  Penalized variance =  2.6140645
Iteration 1:  Penalized variance =  2.2131253
Iteration 2:  Penalized variance =  2.1129517
Iteration 3:  Penalized variance =  2.0984588
Iteration 4:  Penalized variance =  2.0942991
Iteration 5:  Penalized variance =  2.0915686
Iteration 6:  Penalized variance =  2.0855053
Iteration 7:  Penalized variance =  2.0835057
Iteration 8:  Penalized variance =  2.0835037
Iteration 9:  Penalized variance =  2.0835037
Global minimum variance portfolio
Short selling not allowed
Number of obs = 779
Sample: 1955m2 thru 2019m12
─────────────┬────────────
             │     Weight
─────────────┼────────────
         AAA │   .0131033
         AFH │   .3757663
         ARD │   .6108308
         CPH │   .0002995
─────────────┴────────────
Portfolio return    = 0.4278
Portfolio std. dev. = 1.4432
Risk-free rate      = 0.0000
Sharpe ratio        = 0.2964
```

Now the stock CPH appears with a positive weight.

The output of finportfolio with noshort now displays an iteration log. The restriction on short selling places a nonlinear constraint on the optimization problem, so iterative methods must be used to find the optimal portfolio.

◁

▷ Example 4: Target return without short sellling

The target() and noshort options can be combined. There are two important implications when combining these options. First, the portfolio chosen with noshort will have variance at least as high as, and usually higher than, the portfolio chosen without noshort if the unconstrained portfolio involves short selling. Second, without short selling, the target return on the portfolio is bounded above by the highest-return asset in the portfolio.

The four assets from example 3 can be used to illustrate these points. Using returns r_aaa through r_cph, we seek a minimum variance portfolio with a target return of 0.5% monthly:

```
. finportfolio minvariance r_aaa-r_cph, target(0.5)
Minimum variance portfolio with fixed return
Short selling allowed
Number of obs = 779
Sample: 1955m2 thru 2019m12
─────────────┬──────────
             │    Weight
─────────────┼──────────
         AAA │  .2272578
         AFH │  .4851069
         ARD │  .4876142
         CPH │ -.1999789
─────────────┴──────────
Portfolio return    = 0.5000 (fixed return)
Portfolio std. dev. = 1.5201
Risk-free rate      = 0.0000
Sharpe ratio        = 0.3289
```

The portfolio achieves a standard deviation of 1.52% and puts a negative weight on stock CPH. Seeking the same return again but without short selling, we obtain

```
. finportfolio minvariance r_aaa-r_cph, target(0.5) noshort
Iteration 0:  Penalized variance =  2.7490505
Iteration 1:  Penalized variance =  2.4205764
Iteration 2:  Penalized variance =  2.4077978
Iteration 3:  Penalized variance =  2.4035639
Iteration 4:  Penalized variance =   2.401265
Iteration 5:  Penalized variance =  2.3954566
Iteration 6:  Penalized variance =   2.388044
Iteration 7:  Penalized variance =  2.3880425
Iteration 8:  Penalized variance =  2.3880425

Minimum variance portfolio with fixed return
Short selling not allowed

Number of obs = 779
Sample: 1955m2 thru 2019m12
─────────────┬──────────
             │    Weight
─────────────┼──────────
         AAA │  .1620458
         AFH │  .4580245
         ARD │  .3798646
         CPH │  .0000651
─────────────┴──────────

Portfolio return    = 0.5000 (fixed return)
Portfolio std. dev. = 1.5451
Risk-free rate      = 0.0000
Sharpe ratio        = 0.3236
```

which achieves the same 0.5% return, with all positive weights, but a slightly higher portfolio standard deviation of 1.55%.

Seeking an even more aggressive portfolio, we set the monthly return to 0.62% and continue to limit ourselves to positive weights.

```
. finportfolio minvariance r_aaa-r_cph, target(0.62) noshort
Finding initial values...
Iteration 0:  Penalized variance =  4.5106848
Iteration 1:  Penalized variance =  4.4282026
Iteration 2:  Penalized variance =  4.4111366  (backed up)
Iteration 3:  Penalized variance =  4.3989881  (backed up)
Iteration 4:  Penalized variance =  4.3792764
Iteration 5:  Penalized variance =  4.3181569
Iteration 6:  Penalized variance =  4.2592891
Iteration 7:  Penalized variance =  4.2446185
Iteration 8:  Penalized variance =  4.2410388
Iteration 9:  Penalized variance =  4.2406448
Iteration 10: Penalized variance =  4.2405455
Iteration 11: Penalized variance =  4.2405251
Iteration 12: Penalized variance =  4.2405249

Minimum variance portfolio with fixed return
Short selling not allowed

Number of obs = 779
Sample: 1955m2 thru 2019m12
------------------------------
             |      Weight
-------------+----------------
         AAA |   .3933788
         AFH |   .6063672
         ARD |    .000222
         CPH |    .000032
------------------------------

Portfolio return    = 0.6200 (fixed return)
Portfolio std. dev. = 2.0589
Risk-free rate      = 0.0000
Sharpe ratio        = 0.3011
```

More and more weight is being placed on the highest-returning assets in the group, namely, AAA and AFH. The portfolio standard deviation also grows. In going from a target of 0.5% to 0.62%, the Sharpe ratio falls, indicating that risk is being taken on faster than reward to generate the required return.

Increasing the desired return outside the range of observed returns, say, to 0.67% per month, will lead to an error; there is no linear combination of assets with strictly positive coefficients that can generate a return larger than the largest return in the asset group.

◁

Maximum Sharpe ratio portfolios

The methods discussed previously all minimized the variance of the portfolio subject to some constraints. The next methods maximize the Sharpe ratio. The Sharpe ratio is defined as

$$S_p = \frac{r_p - r^f}{\sigma_p}$$

where S_p is the Sharpe ratio of portfolio p, r_p is the average return on the portfolio, r^f is the risk-free rate, and σ_p is the standard deviation of the portfolio. The numerator is the so-called excess return of the portfolio over the risk-free rate, and the denominator captures risk. An increase in the excess return improves the Sharpe ratio; an increase in portfolio standard deviation reduces the Sharpe ratio.

▷ Example 5: The maximum Sharpe ratio portfolio

We next use our four example stocks again to maximize the Sharpe ratio, allowing short selling.

```
. finportfolio maxsharpe r_aaa-r_cph
Maximum Sharpe ratio portfolio
Short selling allowed
Number of obs = 779
Sample: 1955m2 thru 2019m12
```

	Weight
AAA	.3103278
AFH	.5276325
ARD	.427776
CPH	-.2657363

```
Portfolio return    = 0.5297
Portfolio std. dev. = 1.5982
Risk-free rate      = 0.0000
Sharpe ratio        = 0.3315
```

Allowing short selling, the return associated with the maximum Sharpe ratio is 0.53% per month, with a standard deviation of 1.60%; the maximum Sharpe ratio achieved is 0.33. Three of the stocks have a positive weight (AAA, AFH, and ARD). One stock (CPH) has a negative weight.

If we disallow short selling, we obtain

```
. finportfolio maxsharpe r_aaa-r_cph, noshort
Iteration 0:  Penalized Sharpe ratio =  .29721912
Iteration 1:  Penalized Sharpe ratio =  .31860978
Iteration 2:  Penalized Sharpe ratio =  .32122602
Iteration 3:  Penalized Sharpe ratio =   .3218071
Iteration 4:  Penalized Sharpe ratio =   .3230303
Iteration 5:  Penalized Sharpe ratio =  .32347348
Iteration 6:  Penalized Sharpe ratio =  .32347557
Iteration 7:  Penalized Sharpe ratio =  .32347561
Maximum Sharpe ratio portfolio
Short selling not allowed
Number of obs = 779
Sample: 1955m2 thru 2019m12
```

	Weight
AAA	.1855063
AFH	.4702679
ARD	.3433929
CPH	.0008328

```
Portfolio return    = 0.5112
Portfolio std. dev. = 1.5780
Risk-free rate      = 0.0000
Sharpe ratio        = 0.3240
```

The maximum Sharpe ratio is now 0.32. Because we have disallowed short selling, the maximum Sharpe ratio is lower than it was in the unconstrained case.

So far, the implied risk-free return rate was 0. We can change this assumption with the `rfrate()` option, which takes a variable containing the return on a risk-free asset. In our data, we have the federal funds rate, `fedfunds`, which is a benchmark interest rate that is generally considered nearly risk free. It is quoted at an annual rate. We create a new variable with the monthly return on the federal funds rate and use it as the risk-free rate. We continue to disallow short selling.

```
. generate double rf = fedfunds / 12

. finportfolio maxsharpe r_aaa-r_cph, rfrate(rf) noshort
Iteration 0:  Penalized Sharpe ratio =  .04936168
Iteration 1:  Penalized Sharpe ratio =  .06507217  (backed up)
Iteration 2:  Penalized Sharpe ratio =  .06670697  (backed up)
Iteration 3:  Penalized Sharpe ratio =  .06937459
Iteration 4:  Penalized Sharpe ratio =  .07818756
Iteration 5:  Penalized Sharpe ratio =  .10066681
Iteration 6:  Penalized Sharpe ratio =  .10396478
Iteration 7:  Penalized Sharpe ratio =  .10463189
Iteration 8:  Penalized Sharpe ratio =  .10506173
Iteration 9:  Penalized Sharpe ratio =  .10541845
Iteration 10: Penalized Sharpe ratio =  .10647209
Iteration 11: Penalized Sharpe ratio =   .1080777
Iteration 12: Penalized Sharpe ratio =  .10808199
Iteration 13: Penalized Sharpe ratio =  .10808199

Maximum Sharpe ratio portfolio
Short selling not allowed

Number of obs = 779
Sample: 1955m2 thru 2019m12
------------------------------
             |      Weight
-------------+----------------
         AAA |   .1393517
         AFH |   .8603745
         ARD |   .0001329
         CPH |    .000141
------------------------------

Portfolio return    = 0.6459
Portfolio std. dev. = 2.2554
Risk-free rate      = 0.4002 (rf)
Sharpe ratio        = 0.1082
```

Now the goal is to reach the maximum return-to-risk ratio net of the risk-free rate. The optimal weights obtain a Sharpe ratio of 0.11, meaning that for every 1% increase in standard deviation, the optimal portfolio attains an 0.11% increase in monthly return, net of the risk-free rate. The portfolio return is 0.65% monthly, with a standard deviation of 2.26%. This return is close to the maximum return that the constrained portfolio can attain, namely, the return of the best-performing asset, 0.66%.

◁

Portfolios with fixed weights

The previous two methods minimized or maximized a criterion function to arrive at optimal weights. Two methods are available that use fixed portfolio weights and thus do not perform any optimization: equal weights and fixed user-defined weights.

▷ Example 6: A portfolio with equal weights

An equally weighted portfolio simply assigns equal weights to all assets in the portfolio. Continuing with our four example assets, the equally weighted portfolio is

```
. finportfolio equal r_aaa-r_cph
Equally weighted portfolio
Number of obs = 779
Sample: 1955m2 thru 2019m12
```

	Weight
AAA	.25
AFH	.25
ARD	.25
CPH	.25

```
Portfolio return    = 0.4809
Portfolio std. dev. = 1.6167
Risk-free rate      = 0.0000
Sharpe ratio        = 0.2975
```

Despite its simplicity, the equally weighted portfolio can be useful as a benchmark or to explore portfolio characteristics when a simple allocation rule is made. This equally weighted portfolio achieves a monthly return of 0.48% with a standard deviation of 1.62%, for a Sharpe ratio of just under 0.30. These characteristics can serve as a baseline to judge the higher-return, lower-variance portfolios generated by the minimum variance and maximum Sharpe ratio methods.

◁

▷ Example 7: A portfolio with user-defined weights

Finally, custom weights can be assigned using the `fixed` method. The weights, which must sum to one, may be specified using a Stata matrix and are then applied to the assets in the portfolio.

```
. matrix w = (0.3, 0.4, 0.2, 0.1)
. matrix colnames w = r_aaa r_afh r_ard r_cph
. finportfolio fixed r_aaa-r_cph, at(w)
Fixed weights portfolio
Number of obs = 779
Sample: 1955m2 thru 2019m12
```

	Weight
AAA	.3
AFH	.4
ARD	.2
CPH	.1

```
Portfolio return    = 0.5303
Portfolio std. dev. = 1.6861
Risk-free rate      = 0.0000
Sharpe ratio        = 0.3145
```

We create the row vector in the first line and properly label its columns in the second line. We then specify the row vector name in `finportfolio`'s `at()` option. The custom weights are 30% on the AAA stock, 40% on the AFH stock, 20% on the ARD stock, and 10% on the CPH stock. The return generated from these weights is 0.53% monthly, with a standard deviation of 1.69%.

Fixed weights can also be specified in the form *varname*=#. We can type

```
. finportfolio fixed r_aaa-r_cph, at(r_aaa=0.3 r_afh=0.6 r_cph=0.1)
Fixed weights portfolio
Number of obs = 779
Sample: 1955m2 thru 2019m12
```

	Weight
AAA	.3
AFH	.6
ARD	0
CPH	.1

```
Portfolio return    = 0.6059
Portfolio std. dev. = 2.0311
Risk-free rate      = 0.0000
Sharpe ratio        = 0.2983
```

Notice that, because we did not specify a weight for variable `r_ard`, its weight was set to 0.

◁

▷ Example 8: Generating portfolio returns

The `generate()` option generates a new variable that contains the return from the portfolio. Let's compute the equally weighted return to illustrate this option.

```
. finportfolio equal r_aaa-r_cph, generate(p_eweight, label("Equal"))
Equally weighted portfolio
Number of obs = 779
Sample: 1955m2 thru 2019m12
```

	Weight
AAA	.25
AFH	.25
ARD	.25
CPH	.25

```
Portfolio return    = 0.4809
Portfolio std. dev. = 1.6167
Risk-free rate      = 0.0000
Sharpe ratio        = 0.2975
```

Similarly, we can compute the no-short-sales minimum variance portfolio and generate its returns.

```
. finportfolio minvariance r_aaa-r_cph, noshort
> generate(p_minvar, label("MinVar"))
Iteration 0:  Penalized variance =  2.6140645
Iteration 1:  Penalized variance =  2.2131253
Iteration 2:  Penalized variance =  2.1129517
Iteration 3:  Penalized variance =  2.0984588
Iteration 4:  Penalized variance =  2.0942991
Iteration 5:  Penalized variance =  2.0915686
Iteration 6:  Penalized variance =  2.0855053
Iteration 7:  Penalized variance =  2.0835057
Iteration 8:  Penalized variance =  2.0835037
Iteration 9:  Penalized variance =  2.0835037

Global minimum variance portfolio
Short selling not allowed

Number of obs = 779
Sample: 1955m2 thru 2019m12
------------------------------
             |      Weight
-------------+----------------
         AAA |   .0131033
         AFH |   .3757663
         ARD |   .6108308
         CPH |   .0002995
------------------------------

Portfolio return    = 0.4278
Portfolio std. dev. = 1.4432
Risk-free rate      = 0.0000
Sharpe ratio        = 0.2964
```

The two portfolios are now stored in the dataset in variables `p_eweight` and `p_minvar`. We can summarize them:

```
. summarize p_eweight p_minvar
    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
   p_eweight |        779    .4809442    1.616726  -9.104359   6.934312
    p_minvar |        779    .4277795    1.443208  -6.640849   6.721109
```

The minimum variance portfolio has lower standard deviation and also slightly lower return. Notice that the means and standard deviations of the portfolios match the computations from the `finportfolio` output.

◁

Stored results

`finportfolio` stores the following in `r()`:

Scalars

`r(N)`	number of observations
`r(rfrate)`	risk-free rate
`r(target)`	target return, if `target()` specified
`r(short)`	`1` if short selling is allowed, `0` otherwise
`r(tmin)`	minimum time
`r(tmax)`	maximum time
`r(return)`	portfolio return
`r(var)`	portfolio variance
`r(sharpe)`	Sharpe ratio
`r(gamma)`	penalty term `gamma`, if `noshort` was specified
`r(converged)`	`1` if converged, `0` otherwise

Macros

`r(newvar)`	name of variable used to store portfolio returns
`r(method)`	method for selecting portfolio weights

Matrices

`r(b)`	vector of portfolio weights

Methods and formulas

Methods and formulas are presented under the following headings:

Introduction

Throughout, the setting is a collection of K returns, $\mathbf{r} = (r_1, r_2, \ldots, r_K)'$. These returns have $K \times K$ covariance matrix $\mathbf{\Sigma}$. A portfolio is a weighted average of returns. A portfolio with weights $\mathbf{w} = (w_1, w_2, \ldots, w_K)'$ has portfolio return

$$r_w = r_1 w_1 + \cdots + r_K w_K = \mathbf{r}'\mathbf{w}$$

and has portfolio variance

$$\sigma_w^2 = \mathbf{w}'\mathbf{\Sigma}\mathbf{w}$$

The optimization-based portfolio selection techniques are based on either the minimization of a loss criterion or the maximization of a gain criterion.

Minimum variance

The minimum variance method finds the portfolio that minimizes portfolio variance, subject to the constraint that the portfolio weights sum to 1. This is also called the global minimum variance portfolio,

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}\mathbf{w}'\boldsymbol{\Sigma}\mathbf{w} \\ \text{s.t.} \quad & \sum w_i = 1 \\ & \mathbf{r}'\mathbf{w} = r^{\text{target}} \end{aligned}$$

where r^{target} is the target return specified in the `target()` option. This problem has a quadratic objective subject to a linear constraint. The first-order conditions are

$$\begin{pmatrix} \boldsymbol{\Sigma} & \mathbf{E}' \\ \mathbf{E} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{w} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{d} \end{pmatrix}$$

where

$$\mathbf{E} = \begin{pmatrix} \mathbf{r}' \\ \mathbf{1} \end{pmatrix}$$

$$\mathbf{d} = \begin{pmatrix} r^* \\ 1 \end{pmatrix}$$

and $\mathbf{1}$ is a conforming vector of 1s. This is a linear problem; its solution is $\mathbf{w}^*$, the optimal weights.

Minimum variance without short sales

The minimum variance problem can produce solutions with negative portfolio weights, representing assets that the portfolio holder would sell short. Sometimes, a solution is desired that restricts the portfolio to exhibit only nonnegative weights. The formal problem becomes

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}\mathbf{w}'\boldsymbol{\Sigma}\mathbf{w} \\ \text{s.t.} \quad & \sum w_i = 1 \\ & \mathbf{r}'\mathbf{w} = r^{\text{target}} \\ & w_i \geq 0 \qquad \forall i \end{aligned}$$

The `noshort` option accomplishes this by augmenting the standard objective function with a barrier function,

$$\min_{\mathbf{w}} \quad \frac{1}{2}\mathbf{w}'\boldsymbol{\Sigma}\mathbf{w} - \gamma \sum_{i=1}^{K} \ln(w_i)$$

where γ is a tuning parameter. The default value for γ depends on whether a target return is specified. If it is, the default is $\gamma = 0.0001 r^{\text{target}}$. When no target return is specified, $\gamma = 0.0001(r^{\text{min}} + r^{\text{max}})/2$, where r^{min} and r^{max} are, respectively, the minimum and maximum return values. The result is an approximate solution to the inequality-constrained problem.

Maximum Sharpe ratio

The Sharpe ratio is the ratio of portfolio return (net of a risk-free rate, r^f) to portfolio standard deviation,

$$S(\mathbf{w}) = \frac{\mathbf{r}'\mathbf{w} - r^f}{\sigma(\mathbf{w})}$$

The maximum Sharpe ratio method finds the portfolio weights **w** that maximize the Sharpe ratio.

Maximum Sharpe ratio without short sales

The maximum Sharpe ratio method can produce some weights that are negative, representing assets that the portfolio holder would like to sell short. The `noshort` option restricts the portfolio weights to be nonnegative. This is accomplished by augmenting the objective with a barrier function,

$$\max_{\mathbf{w}} \quad \frac{\mathbf{r}'\mathbf{w} - r^f}{\sigma(\mathbf{w})} + \gamma \sum_{i=1}^{K} \ln(w_i)$$

where γ is a tuning parameter, $\gamma = 0.0001(r^{\min} + r^{\max})/2$, and $r^{\min}$ and $r^{\max}$ are, respectively, the minimum and maximum return values. The solution to the augmented problem guarantees that portfolio weights will all be nonnegative.

References

Campbell, J. Y. 2018. *Financial Decisions and Markets: A Course in Asset Pricing*. Princeton, NJ: Princeton University Press.

Hurn, S., V. L. Martin, P. C. B. Phillips, and J. Yu. 2020. *Financial Econometric Modeling*. Oxford: Oxford University Press.

Also see

[FIN] **finreturns** — Generate financial returns+

finregress capm — Capital asset pricing model (CAPM)+

+This command is part of StataNow.

Description

`finregress capm` fits a time-series capital asset pricing model (CAPM), which relates assets' characteristics to risk factors of interest. A collection of dependent variables, usually asset returns, is regressed on a common set of independent variables, usually including a benchmark return (an overall market return) and possibly additional factors. Both the independent and dependent variables can be adjusted for a risk-free return. Coefficients for all assets are estimated simultaneously, allowing for joint tests across assets.

Quick start

Regress returns `r1`, `r2`, and `r3` on a market portfolio return `rmkt`

```
finregress capm r1 r2 r3 = rmkt
```

Same as above, but adjust dependent variables by subtracting risk-free rate `rf`

```
finregress capm r1 r2 r3 = rmkt, rfrate(rf)
```

Same as above, but adjust independent variable `rmkt` for the risk-free rate

```
finregress capm r1 r2 r3 = rmkt, rfrate(rf) adjust
```

Three-factor CAPM, adjusting `rmkt` but not variable `x1` or `x2`

```
finregress capm r1 r2 r3 = rmkt x1 x2, rfrate(rf) adjust(rmkt)
```

Same as above, but include an economic recession indicator `rec`

```
finregress capm r1 r2 r3 = rmkt x1 x2 i.rec, rfrate(rf) adjust(rmkt)
```

Menu

Statistics > Financial statistics > Capital asset pricing model (CAPM)

Syntax

Basic syntax

```
finregress capm depvars [= [indepvars] ] [if] [in] [weight] [, options]
```

Syntax with a risk-free rate

```
finregress capm depvars [= [indepvars] ] [if] [in] [weight] , rfrate(varname) [options]
```

options	Description
Model	
`rfrate(`*varname*`)`	specify risk-free rate variable and adjust all *depvars*
`adjust`	adjust all independent variables by risk-free rate variable
`adjust(`*varlist*`)`	adjust independent variables in *varlist* by risk-free rate variable
`noconstant`	omit constant term
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be `robust`, `unadjusted`, `cluster` *clustvar*, or `hac` *hacspec*
Reporting	
`level(`#`)`	set confidence level; default is `level(95)`
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
`coeflegend`	display legend instead of statistics

You must `tsset` your data before using `finregress capm`; see [TS] **tsset**.

indepvars may contain factor variables; see **[U] 11.4.3 Factor variables**.

depvars, *indepvars*, *varname*, and *varlist* may contain time-series operators; see **[U] 11.4.4 Time-series varlists**.

`collect` is allowed; see **[U] 11.1.10 Prefix commands**.

`aweights` and `fweights` are allowed; see **[U] 11.1.6 weight**.

`coeflegend` does not appear in the dialog box.

See **[U] 20 Estimation and postestimation commands** for more capabilities of estimation commands.

Options

Model

`rfrate(`*varname*`)` specifies the risk-free rate used in the CAPM. If `rfrate()` is not specified, no adjustment is made, and the risk-free rate is implicitly set to 0 in all periods. When `rfrate()` is specified, it indicates that you wish to transform all dependent variables by subtracting the same risk-free rate, the variable in `rfrate()`.

`adjust` and `adjust(`*varlist*`)` specify which independent variables are to be adjusted by subtracting the risk-free rate in `rfrate()`. `adjust` adjusts all independent variables. `adjust(`*varlist*`)` adjusts only the independent variables specified in *varlist*.

`noconstant` omits the constants in the CAPM.

SE/Robust

vce(*vcetype*) specifies the type of standard error reported, which includes types that are robust to some kinds of misspecification (robust) and that allow for intragroup correlation (cluster *clustvar*); see [R] ***vce_option***.

vce(unadjusted) requests conventional standard errors appropriate under homoskedasticity and no autocorrelation. See [R] **gmm** for details.

vce(hac *hacspec*) requests a heteroskedasticity- and autocorrelation-consistent (HAC) variance–covariance matrix. The full syntax of *hacspec* is one of the following:

vce(hac *kernel* [#]) requests a HAC variance–covariance matrix using the specified kernel (see below) with optional # lags. The bandwidth of a kernel is equal to # + 1. If # is not specified, a kernel with $N - 2$ lags is used, where N is the sample size.

vce(hac *kernel* opt [#]) requests a HAC variance–covariance matrix using the specified kernel (see below), and the lag order is selected using Newey and West's (1994) optimal lag-selection algorithm. # is an optional tuning parameter that affects the lag order selected; see the discussion in *Methods and formulas* in [R] **ivregress**.

kernel may be one of the following:

bartlett or nwest requests the Bartlett (Newey–West) kernel.

parzen or gallant requests the Parzen (Gallant 1987) kernel.

quadraticspectral or andrews requests the quadratic spectral (Andrews 1991) kernel.

Reporting

level(#); see [R] **Estimation options**.

display_options: noci, nopvalues, noomitted, vsquish, noemptycells, baselevels, allbaselevels, nofvlabel, fvwrap(#), fvwrapon(*style*), cformat(%*fmt*), pformat(%*fmt*), sformat(%*fmt*), and nolstretch; see [R] **Estimation options**.

The following option is available with finregress capm but is not shown in the dialog box:

coeflegend; see [R] **Estimation options**.

Remarks and examples

A CAPM relates asset characteristics, usually returns, to independent variables, often called risk factors or simply factors. These regressions can be run asset by asset or fit jointly on a collection of returns.

The classic CAPM relates the returns of each asset i at time t, r_{it}, to the overall market return, r_t^m. The rate of return one may obtain without incurring any risk, r_t^f, is also relevant because only returns in excess of this risk-free rate are considered gains. The model is

$$r_{it} - r_t^f = \alpha_i + \beta_i(r_t^m - r_t^f) + e_{it}$$

The coefficient β_i is called the market beta of asset i. Mechanically, β is a regression coefficient like any other. It has the standard interpretation: if the market return rises by one unit, then the stock's return is expected to rise by β units. Several values of β have special interpretation.

- $\beta < 0$ indicates that the stock moves against the market.
- $\beta = 0$ indicates that the stock's return is unrelated to the market return.
- $0 < \beta < 1$ indicates that the stock's return fluctuates in the same direction as the market but less than one for one.
- $\beta = 1$ indicates that the stock's return fluctuates one for one with the market return.
- $\beta > 1$ indicates that the stock's return varies more than one for one with the market.

The intercept α_i represents expected returns that are not captured by the independent variables in the model. The main assumption of the CAPM is that the intercept terms α_i's are zero. A test of this assumption is carried out by the Gibbons–Ross–Shanken procedure.

▷ Example 1: Single-factor CAPM

We use our fictional financial prices dataset and describe the first few variables.

```
. use https://www.stata-press.com/data/r19/finex
(Fictional stock price data)
. describe datestr-wgt
```

Variable name	Storage type	Display format	Value label	Variable label
datestr	str11	%11s		String date
datem	int	%tm		Monthly date
sp500	double	%10.0g		S&P 500
vol	float	%9.0g		Volatility index
fedfunds	float	%9.0g		Federal funds rate
acme	float	%9.0g		Aciron Medical, Inc.
bat	float	%9.0g		Boron Advanced Technologies
iron	float	%9.0g		Industrial Operations Network
dune	float	%9.0g		Digital Urban Network Enterprise
tyr	float	%9.0g		Tyndale Resources Group
glo	float	%9.0g		Green Logistics, Inc.
spa	float	%9.0g		Space Rocket MFG
wgt	float	%9.0g		Widget Gadgets

Because the dataset contains prices and `finregress` works on returns, we use `finreturns` to convert price series into monthly log returns. We multiply by 100 so that these returns are approximately percentage changes.

```
. quietly finreturns acme-tyr, log(lnr_) multiply(100)
```

Next, we use `finreturns` to calculate monthly log returns for the stock market (`sp500`), which will serve as our independent variable or overall market factor. The question of interest is how excess returns on individual assets vary with excess returns on this benchmark.

```
. quietly finreturns sp500, log(lnr_mkt) multiply(100)
```

To measure excess returns, we use the return in excess of a risk-free rate. The `fedfunds` variable contains the annualized federal funds interest rate and serves as our risk-free rate. Because the asset returns are all measured in monthly log changes, we approximate the monthly risk-free return by constructing the annual log risk-free return and dividing by 12.

```
. generate double rf = 100 * log(1 + fedfunds/100)/12
```

With this preprocessing complete, we are ready to fit a CAPM relating individual excess log stock returns to the excess log return on the market.

Data come in many formats. If our dependent and independent variables were already adjusted for risk-free rate `rf`, we would fit the CAPM by typing

```
. finregress capm lnr_acme-lnr_tyr = lnr_mkt
```

In our case, however, the variables have not been adjusted, so we type

```
. finregress capm lnr_acme-lnr_tyr = lnr_mkt, rfrate(rf) adjust
Capital asset pricing model
Sample: 1955m2 thru 2019m12                          Number of obs = 779
```

	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
lnr_acme						
lnr_mkt	.1316993	.0102147	12.89	0.000	.111679	.1517197
_cons	.0322367	.0378174	0.85	0.394	-.041884	.1063575
lnr_bat						
lnr_mkt	1.530731	.0114377	133.83	0.000	1.508314	1.553148
_cons	.0760026	.0416309	1.83	0.068	-.0055925	.1575977
lnr_iron						
lnr_mkt	1.871524	.012273	152.49	0.000	1.847469	1.895579
_cons	.0556327	.0454103	1.23	0.221	-.0333698	.1446352
lnr_dune						
lnr_mkt	1.878887	.0123726	151.86	0.000	1.854637	1.903137
_cons	.0895071	.046235	1.94	0.053	-.0011119	.1801261
lnr_tyr						
lnr_mkt	1.087576	.0117232	92.77	0.000	1.064599	1.110553
_cons	.0282078	.0443328	0.64	0.525	-.0586828	.1150984

```
Notes: Dependent variables adjusted for risk-free rate rf.
       Independent variable lnr_mkt adjusted for risk-free rate rf.
```

The `rfrate()` option specifies which variable our asset returns are in "excess" of. When `rfrate()` is specified, dependent variables (in this case `lnr_acme-lnr_tyr`) are transformed prior to estimation, generating excess returns r^e_{it} as

$$r^e_{it} = r_{it} - r^f_t$$

In addition, by specifying `adjust`, we also transformed `lnr_mkt` prior to estimation by subtraction of the same risk-free rate variable. Thus, the regression actually run, for asset i, is

$$(r_{it} - r^f_t) = \alpha_i + \beta_i(r^m_t - r^f_t) + e_{it}$$

Turning to the output, we see that each block in the output table corresponds to one asset return. Two coefficients were estimated for each return series: a coefficient on the market and an intercept. All the chosen assets covary positively with the market, with response coefficients ranging from 0.13 to 1.88. The units on both sides of the equation are monthly excess log returns, so the interpretation is that a one-unit change in excess log return in the market is associated with an expected β_i change in excess log return in each asset, measured monthly. The intercept can be interpreted as the excess log return in that asset when the excess log return on the market is 0. For example, the ACME stock (equation `lnr_acme`) has an intercept of about 0.03, indicating that even when excess log market return is 0, the ACME stock still generates an excess monthly log return of about 0.03 or approximately 3 basis points.

◁

There is no requirement that the CAPM uses a single independent variable or factor. Indeed, multifactor models have become popular; see Fama and French (1992, 1996) and Campbell, Lo, and MacKinlay (1997, chap. 6).

▷ Example 2: Multifactor CAPM

Continuing with the previous example, we add the `vol` factor as an independent variable to the model for our five assets. This factor is a return on a fictional volatility index. We use the volatility index as a control, and we do not want to adjust it for the risk-free rate.

```
. finregress capm lnr_acme-lnr_tyr = lnr_mkt vol, rfrate(rf) adjust(lnr_mkt)
Capital asset pricing model
Sample: 1955m2 thru 2019m12                          Number of obs = 779
```

	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
lnr_acme						
lnr_mkt	.1483932	.0119328	12.44	0.000	.1250054	.171781
vol	-.0966061	.0356362	-2.71	0.007	-.1664517	-.0267604
_cons	-.0086329	.0419574	-0.21	0.837	-.0908679	.0736021
lnr_bat						
lnr_mkt	1.52412	.0131144	116.22	0.000	1.498417	1.549824
vol	.0382555	.0400207	0.96	0.339	-.0401837	.1166947
_cons	.0921868	.0450832	2.04	0.041	.0038254	.1805482
lnr_iron						
lnr_mkt	1.851205	.0145758	127.01	0.000	1.822636	1.879773
vol	.1175864	.0443886	2.65	0.008	.0305864	.2045865
_cons	.1053782	.0479793	2.20	0.028	.0113406	.1994159
lnr_dune						
lnr_mkt	1.872238	.014166	132.16	0.000	1.844473	1.900003
vol	.0384778	.0399057	0.96	0.335	-.039736	.1166915
_cons	.1057853	.049148	2.15	0.031	.0094571	.2021135
lnr_tyr						
lnr_mkt	1.083229	.0131535	82.35	0.000	1.057448	1.109009
vol	.0251582	.0392261	0.64	0.521	-.0517235	.1020398
_cons	.0388511	.0483987	0.80	0.422	-.0560086	.1337109

```
Notes: Dependent variables adjusted for risk-free rate rf.
       Independent variable lnr_mkt adjusted for risk-free rate rf.
```

The volatility factor appears with coefficients ranging from −0.10 on the lnr_acme return to 0.12 on the lnr_iron return. The results from these regressions provide information on the level and distribution of the coefficients across assets. We can characterize which assets covary strongly with particular independent variables. One key prediction of the CAPM that is not tested here is whether the coefficients themselves vary systematically with average returns. To investigate this question, see [FIN] **finregress fmb**.

◁

Stored results

finregress capm stores the following in e():

Scalars

e(N)	number of observations
e(k_dv)	number of dependent variables
e(k_indepvars)	number of independent variables, not including the constant
e(tmin)	minimum time
e(tmax)	maximum time

Macros

e(cmd)	finregress
e(subcmd)	capm
e(cmdline)	command as typed
e(depvar)	names of dependent variables
e(indepvars_unadj)	names of unadjusted independent variables
e(indepvars_adj)	names of adjusted independent variables
e(wtype)	weight type
e(wexp)	weight expression
e(title)	title in estimation output
e(vcetype)	title used to label Std. err.
e(vce)	*vcetype* specified in vce()
e(rfrate)	risk-free rate variable, if specified
e(tmaxs)	formatted maximum time
e(tmins)	formatted minimum time
e(tvar)	time variable
e(properties)	b V
e(estat_cmd)	program used to implement estat
e(predict)	program used to implement predict
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved

Matrices

e(b)	coefficient vector
e(V)	variance–covariance matrix of the estimator
e(alpha)	vector of intercepts

Functions

e(sample)	marks estimation sample

In addition to the above, the following is stored in r():

Matrices

r(table)	matrix containing the coefficients with their standard errors, test statistics, *p*-values, and confidence intervals

Note that results stored in r() are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

Let r_{it} be the return for asset i and time t, where $i = 1, 2, \ldots, K$ and $t = 1, 2, \ldots, T$. The CAPM relates r_{it} to the overall market return, r_t^m, while adjusting for the risk-free rate, r_t^f.

We write

$$r_{it} - r_t^f = \alpha_i + \beta_i(r_t^m - r_t^f) + e_{it}$$

for the single-factor CAPM and

$$r_{it} - r_t^f = \alpha_i + \beta_{1i}(x_{1t} - r_t^f) + \beta_{2i}(x_{2t} - r_t^f) + \cdots + e_{it}$$

for the multifactor CAPM. α_i's and the β's are estimated parameters, and e_{it} is the error. Notice the i-index on the coefficients; each asset return series is allowed to have its own intercept and beta. In many applications, one of the independent variables is the excess return on a benchmark or market index; the coefficient on this variable is referred to as an asset's market beta. The intercept, which represents average excess returns that are not captured by the independent variables, is called Jensen's alpha.

The CAPM can also be thought of as a factor structure on the behavior of returns. Let $\mathbf{r}_t$ be an $K \times 1$ column vector of (adjusted) returns. Let $\mathbf{f}_t$ be an $M \times 1$ column vector of (adjusted) factors. Note that these factors are observed independent variables in this case. Then the CAPM is

$$\mathbf{r}_t = \boldsymbol{\alpha} + \boldsymbol{\Lambda}\mathbf{f}_t + \mathbf{e}_t$$

where $\boldsymbol{\alpha}$ is an $K \times 1$ vector of intercept terms, $\boldsymbol{\Lambda}$ is an $K \times M$ matrix of coefficients, and $\mathbf{e}_t$ is an $K \times 1$ vector of error terms.

The collection of time-series regressions is estimated jointly by the generalized method of moments; see [R] **gmm** for details.

References

Andrews, D. W. K. 1991. Heteroskedasticity and autocorrelation consistent covariance matrix estimation. *Econometrica* 59: 817–858. https://doi.org/10.2307/2938229.

Campbell, J. Y., A. W. Lo, and A. C. MacKinlay. 1997. *The Econometrics of Financial Markets*. Princeton, NJ: Princeton University Press. https://doi.org/10.2307/j.ctt7skm5.

Fama, E. F., and K. R. French. 1992. The cross-section of expected stock returns. *Journal of Finance* 47: 427–465. https://doi.org/10.1111/j.1540-6261.1992.tb04398.x.

———. 1996. Multifactor explanations of asset pricing anomalies. *Journal of Finance* 51: 55–84. https://doi.org/10.1111/j.1540-6261.1996.tb05202.x.

Gallant, A. R. 1987. *Nonlinear Statistical Models*. New York: Wiley. https://doi.org/10.1002/9780470316719.

Newey, W. K., and K. D. West. 1994. Automatic lag selection in covariance matrix estimation. *Review of Economic Studies* 61: 631–653. https://doi.org/10.2307/2297912.

Also see

[FIN] **finregress capm postestimation** — Postestimation tools for finregress capm+

[FIN] **finportfolio** — Financial portfolio selection+

[FIN] **finregress fmb** — Fama–MacBeth regression+

[FIN] **finreturns** — Generate financial returns+

[U] **20 Estimation and postestimation commands**

finregress capm postestimation — Postestimation tools for finregress capm[+]

[+]Postestimation features after `finregress capm` are part of StataNow.

Postestimation commands

The following postestimation commands are of special interest after `finregress capm`:

Command	Description
`estat grstest`	Gibbons–Ross–Shanken test

The following standard postestimation commands are also available:

Command	Description
`estat summarize`	summary statistics for the estimation sample
`estat vce`	variance–covariance matrix of the estimators (VCE)
`estimates`	cataloging estimation results
`etable`	table of estimation results
`lincom`	point estimates, standard errors, testing, and inference for linear combinations of parameters
`nlcom`	point estimates, standard errors, testing, and inference for nonlinear combinations of parameters
`predict`	linear predictions and residuals
`predictnl`	point estimates, standard errors, testing, and inference for generalized predictions
`test`	Wald tests of simple and composite linear hypotheses
`testnl`	Wald tests of nonlinear hypotheses

predict

Description for predict

`predict` creates a new variable containing predictions such as linear predictions and residuals.

Menu for predict

Statistics > Postestimation

Syntax for predict

`predict` [*type*] *newvar* [*if*] [*in*] [, *statistic* `equation(`*eqspec*`)`]

statistic	Description
Main	
`xb`	linear prediction; the default
`residuals`	residuals

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

Options for predict

Main

`xb`, the default, calculates the linear prediction for the specified equation. Note that the linear prediction for `finregress capm` models is computed in terms of returns, rather than excess returns. That is, $r_t^f + \mathbf{x}_t'\widehat{\mathbf{b}}$, where $\mathbf{x}_t'\widehat{\mathbf{b}}$ is the typical linear prediction of the regression model, and r_t^f is the risk-free rate specified in the `rfrate()` option of `finregress capm`. See *Methods and formulas* for details.

`residuals` calculates the residuals for the specified equation. The residuals for `finregress capm` models are computed, as usual, by subtracting linear predictions from observed values. But keep in mind that the linear prediction for `finregress capm` models is computed in terms of returns, rather than excess returns. That is, $r_t^f + \mathbf{x}_t'\widehat{\mathbf{b}}$, where $\mathbf{x}_t'\widehat{\mathbf{b}}$ is the typical linear prediction of the regression model, and r_t^f is the risk-free rate specified in the `rfrate()` option of `finregress capm`. See *Methods and formulas* for details.

`equation(`*eqspec*`)` specifies to which equation you are referring. *eqspec* can be an equation number or an equation name. For example, if *eqspec* is a number, then `equation(#1)` would mean that the calculation is to be made for the first equation (that is, for the first dependent variable), `equation(#2)` would mean the second, and so on. If *eqspec* is a name, then `equation(return)` would refer to the equation named `return`, and `equation(r_asset)` would refer to the equation named `r_asset`.

If you do not specify `equation()`, the results are the same as if you had specified `equation(#1)`.

estat grstest

Description for estat grstest

`estat grstest` performs a Gibbons–Ross–Shanken test on the intercepts that are estimated simultaneously from a financial regression with multiple assets. The null hypothesis is that those intercepts are jointly zero.

Menu for estat

Statistics > Postestimation

Syntax for estat grstest

```
estat grstest [ , finite ]
```

`collect` is allowed; see **[U] 11.1.10 Prefix commands**.

Option for estat grstest

`finite` performs the test as described in Gibbons, Ross, and Shanken (1989), which assumes normality of the errors and uses a finite-sample F statistic.

Remarks and examples

▷ Example 1: The Gibbons–Ross–Shanken test

We continue with example 1 of [FIN] **finregress capm**. We have fit a CAPM for the excess returns of five assets on excess market return.

```
. use https://www.stata-press.com/data/r19/finex
(Fictional stock price data)
. quietly finreturns acme-tyr, log(lnr_) multiply(100)
. quietly finreturns sp500, log(lnr_mkt) multiply(100)
. generate double rf = 100 * log(1 + fedfunds/100)/12
. finregress capm lnr_acme-lnr_tyr = lnr_mkt, rfrate(rf) adjust
Capital asset pricing model
Sample: 1955m2 thru 2019m12                          Number of obs = 779
```

	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
lnr_acme						
lnr_mkt	.1316993	.0102147	12.89	0.000	.111679	.1517197
_cons	.0322367	.0378174	0.85	0.394	-.041884	.1063575
lnr_bat						
lnr_mkt	1.530731	.0114377	133.83	0.000	1.508314	1.553148
_cons	.0760026	.0416309	1.83	0.068	-.0055925	.1575977
lnr_iron						
lnr_mkt	1.871524	.012273	152.49	0.000	1.847469	1.895579
_cons	.0556327	.0454103	1.23	0.221	-.0333698	.1446352
lnr_dune						
lnr_mkt	1.878887	.0123726	151.86	0.000	1.854637	1.903137
_cons	.0895071	.046235	1.94	0.053	-.0011119	.1801261
lnr_tyr						
lnr_mkt	1.087576	.0117232	92.77	0.000	1.064599	1.110553
_cons	.0282078	.0443328	0.64	0.525	-.0586828	.1150984

```
Notes: Dependent variables adjusted for risk-free rate rf.
       Independent variable lnr_mkt adjusted for risk-free rate rf.
```

In many asset-pricing applications, the key test is not for the coefficient on the market return in each time-series regression. Instead, the question of interest is whether the independent variables explain all the variation in the average level of the dependent variables. Statistically, the key implication is that the intercepts of the regressions should be small. An intercept other than zero indicates that there is still some average return that is not explained by the asset's relationship with the independent variables.

It is then natural to test for whether the intercepts are jointly zero. Because this is a multiequation regression estimated simultaneously, the joint test of significance is just a Wald test. We could use the `test` command to perform such tests, but instead let's use the `estat grstest` command after `finregress capm` to more conveniently perform the test.

```
. estat grstest

Gibbons-Ross-Shanken test
H0: All intercept terms are zero

  No. of dependent vars. =      5
No. of independent vars. =      1
     No. of time periods =    779

                 chi2(5) =  6.662
          Prob > chi2 = 0.2470
```

We do not find evidence that the intercepts are jointly different from zero.

Gibbons, Ross, and Shanken (1989) developed a version of the test that assumes normality of the errors and uses a finite-sample F statistic. This statistic can be requested with the `finite` option.

```
. estat grstest, finite

Gibbons-Ross-Shanken test
H0: All intercept terms are zero

  No. of dependent vars. =      5
No. of independent vars. =      1
     No. of time periods =    779

              F(5, 773) =  1.321
               Prob > F = 0.2532
```

Results are similar. We fail to reject the null hypothesis that the intercept is jointly zero for all assets.

◁

Stored results

`estat grstest` stores the following in `r()`:

Scalars

`r(N)`	number of observations
`r(k_dv)`	number of dependent variables
`r(k_indepvars)`	number of independent variables
`r(df)`	degrees of freedom
`r(chi2)`	χ^2
`r(p)`	p-value

`estat grstest` with the `finite` option stores the following in `r()`:

Scalars

`r(N)`	number of observations
`r(k_dv)`	number of dependent variables
`r(k_indepvars)`	number of independent variables
`r(F)`	F statistic
`r(df)`	denominator degrees of freedom
`r(df_r)`	numerator degrees of freedom
`r(p)`	p-value

Methods and formulas

Methods and formulas are presented under the following headings:

predict

`predict` with the `xb` option provides the linear prediction. With the `residuals` option, `predict` calculates the errors of the linear prediction.

When the `rfrate()` option is specified in the `finregress capm` command, `predict` with the `xb` option produces the linear prediction plus the risk-free rate specified in `rfrate()`. If the asset return is r_{it} and the risk-free rate is r_t^f, then the prediction is

$$\hat{r}_{it} = r_t^f + \mathbf{x}_t'\hat{\mathbf{b}}$$

The linear prediction is therefore in terms of returns, rather than excess returns, because `predict` automatically adds back the risk-free rate.

When some independent variables are also adjusted for the risk-free rate, say, the first l of the M independent variables, then the linear prediction is

$$\hat{r}_{it} = r_t^f + (x_{1t} - r_t^f)\hat{b}_1 + \cdots + (x_{lt} - r_t^f)\hat{b}_l + x_{l+1,t}\hat{b}_{l+1} + \cdots + x_{Mt}\hat{b}_M$$

Notice that the risk-free rate will also affect the residual series we obtain with the `residuals` option of `predict`, because the residuals are simply the observed outcomes minus the linear prediction we defined above.

estat grstest

`estat grstest` performs a test of all intercepts being jointly zero. By default, this is a standard Wald test.

`estat grstest` with the `finite` option implements the Gibbons, Ross, and Shanken (1989, 1167) procedure. Let r_{it} be a collection of K dependent variables (asset returns), and let $\mathbf{x}_t$ be a collection of M independent variables (factors). Observations are over T periods, $t = 1, \ldots, T$. The test for mean-variance efficiency of the independent variables is that in a regression

$$r_{it} = a_i + \mathbf{x}_t'\mathbf{b}_i + \eta_t \qquad \forall i = 1, \ldots, K$$

all the intercepts will be zero, that is,

$$H_0\colon a_i = 0 \qquad \forall i = 1, \ldots, K$$

Let $\mathbf{r}_t$ be the $K \times 1$ vector of returns in period t, let $\hat{\mathbf{a}}$ be the $K \times 1$ vector of estimated intercept terms, and let $\widehat{\mathbf{B}}$ be the $K \times M$ matrix of estimated coefficients on the M independent variables $\mathbf{x}_t$. Then the required pieces for a Wald test are

$$\hat{\boldsymbol{\eta}} = \mathbf{r}_t - \hat{\mathbf{a}} - \widehat{\mathbf{B}}\mathbf{x}_t \qquad (K \times 1)$$

$$\bar{\mathbf{x}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{x}_t \qquad (M \times 1)$$

$$\boldsymbol{\Omega} = \frac{1}{T}\sum_{t=1}^{T}(\mathbf{x}_t - \bar{\mathbf{x}})(\mathbf{x}_t - \bar{\mathbf{x}})' \qquad (M \times M)$$

$$\widehat{\boldsymbol{\Sigma}} = \frac{1}{T-K-1}\sum_{t=1}^{T}\hat{\boldsymbol{\eta}}_t\hat{\boldsymbol{\eta}}_t' \qquad (K \times K)$$

The test statistic formed is

$$\widehat{W} = \frac{T(T-K-M)}{K(T-M-1)}\left(1 + \bar{\mathbf{x}}'\boldsymbol{\Omega}^{-1}\bar{\mathbf{x}}\right)^{-1}\left(\hat{\mathbf{a}}'\widehat{\boldsymbol{\Sigma}}^{-1}\hat{\mathbf{a}}\right)$$

When the residuals are distributed normally with covariance matrix $\boldsymbol{\Sigma}$, Kamstra and Shi (2021) show that the test statistic $\widehat{W} \sim F(K, T-K-M)$.

References

Gibbons, M. R., S. A. Ross, and J. Shanken. 1989. A test of the efficiency of a given portfolio. *Econometrica* 57: 1121–1152. https://doi.org/10.2307/1913625.

Kamstra, M. J., and R. Shi. 2021. A note on the GRS test. Working Paper 202111, Department of Economics, University of Californa, Riverside. https://economics.ucr.edu/repec/ucr/wpaper/202111.pdf.

Also see

[FIN] **finregress capm** — Capital asset pricing model (CAPM)+

[U] 20 Estimation and postestimation commands

finregress fmb — Fama–MacBeth regression+

+This command is part of StataNow.

Description

`finregress fmb` performs Fama and MacBeth (1973) regression, which explores cross-sectional behavior of average returns. This is a two-step regression method. The first step consists of time-series regressions of a set of dependent variables on a set of independent variables (also called factors). These regressions produce coefficients called factor exposures. The second step regresses the cross-section of dependent variables at a point in time against the factor exposures to characterize how the dependent variables vary with factor exposure. The second step is repeated in the cross-section for each time period, and the results are combined over time to produce one number. This combined result is the price of risk, which characterizes how the cross-section of the dependent variables is related to the factor exposures.

Quick start

Find price of risk for assets `r1`, `r2`, and `r3` using market rate `rmkt`

```
finregress fmb r1 r2 r3 = rmkt
```

Same as above, but use excess returns net of the risk-free rate (`rf`) for dependent variables

```
finregress fmb r1 r2 r3 = rmkt, rfrate(rf)
```

Same as above, but adjust the independent variable `rmkt` for the risk-free rate

```
finregress fmb r1 r2 r3 = rmkt, rfrate(rf) adjust
```

Three-factor Fama–MacBeth regression, adjusting all dependent variables and independent variable `rmkt` but not the independent variables `x1` and `x2`

```
finregress fmb r1 r2 r3 = rmkt x1 x2, rfrate(rf) adjust(rmkt)
```

Store the period-by-period price of risk estimates in variables with prefix `lambda`

```
finregress fmb r1 r2 r3 = rmkt x1 x2, rfrate(rf) adjust(rmkt) generate(lambda*)
```

Menu

Statistics > Financial statistics > Fama–MacBeth regression

Syntax

Basic syntax

```
finregress fmb depvars = [ indepvars ] [ if ] [ in ] [ , options ]
```

Syntax with a risk-free rate

```
finregress fmb depvars = [ indepvars ] [ if ] [ in ], rfrate(varname) [ options ]
```

options	Description
Model	
`rfrate(`*varname*`)`	specify risk-free rate variable and adjust all *depvars*
`adjust`	adjust all independent variables by risk-free rate variable
`adjust(`*varlist*`)`	adjust independent variables in *varlist* by risk-free rate variable
`noconstant`	omit the constants in the first-stage regressions
`small`	report small-sample t statistics
`dfk`	use small-sample degrees-of-freedom adjustment
`gls`	use generalized least squares in cross-sectional stage
`generate(`*newvars* | *stub*`*)`	generate variables containing price-of-risk estimates
`shanken`	apply Shanken correction
Reporting	
`level(#)`	set confidence level; default is `level(95)`
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
`coeflegend`	display legend instead of statistics

You must `tsset` your data before using `finregress fmb`; see [TS] **tsset**.

indepvars may contain factor variables; see **[U] 11.4.3 Factor variables**.

depvars, *indepvars*, *varname*, and *varlist* may contain time-series operators; see **[U] 11.4.4 Time-series varlists**.

`collect` is allowed; see **[U] 11.1.10 Prefix commands**.

`coeflegend` does not appear in the dialog box.

See **[U] 20 Estimation and postestimation commands** for more capabilities of estimation commands.

Options

Model

`rfrate(`*varname*`)` specifies the risk-free rate to be used in the first-stage time-series regressions. If `rfrate()` is not specified, no adjustment is made, and the risk-free rate is implicitly set to 0 in all periods. When `rfrate()` is specified, it indicates that you wish to transform all dependent variables by subtracting the same risk-free rate, the variable in `rfrate()`.

`adjust` and `adjust(`*varlist*`)` specify which independent variables are to be adjusted by subtracting the risk-free rate in `rfrate()`. `adjust` adjusts all independent variables. `adjust(`*varlist*`)` adjusts only the independent variables specified in *varlist*.

`noconstant` omits a constant from the first-stage regressions.

`small` specifies that small-sample t statistics be computed for the tests of coefficients instead of large-sample normal z statistics.

`dfk` specifies that a small-sample degrees-of-freedom adjustment (Campbell, Lo, and MacKinlay 1997) be made when estimating the variance of the price-of-risk coefficient estimate. Specifically, $1/T(T-1)$ is used instead of the large-sample divisor $1/T^2$, where T is the number of time periods.

`gls` uses generalized least squares instead of ordinary least squares when computing the price-of-risk coefficient estimate.

`generate(`*newvars* | *stub*`*)` generates new variables containing the period-by-period estimates of the price-of-risk coefficient. The price-of-risk coefficient reported by the Fama–MacBeth regression is the average of these period-by-period estimates. *newvars* must have the same number of variables as are in *depvars*.

`shanken` applies the Shanken (1992) errors-in-variables correction to the variance of the price-of-risk coefficient estimate.

Reporting

`level(#)`; see [R] **Estimation options**.

display_options: `noci`, `nopvalues`, `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`, `nofvlabel`, `fvwrap(#)`, `fvwrapon(`*style*`)`, `cformat(%`*fmt*`)`, `pformat(%`*fmt*`)`, `sformat(%`*fmt*`)`, and `nolstretch`; see [R] **Estimation options**.

The following option is available with `finregress fmb` but is not shown in the dialog box:

`coeflegend`; see [R] **Estimation options**.

Remarks and examples

Remarks are presented under the following headings:

Introduction
The Fama–MacBeth regression

Introduction

`finregress fmb` fits cross-sectional asset pricing models using the method of Fama and MacBeth (1973). In a cross-sectional asset pricing model, we are interested in learning how exposure of asset returns to factors is related to the average performance of those returns. Intuitively, assets that are "more risky" ought to bring higher average returns (otherwise, no one would hold the riskier assets), and the cross-sectional approach provides a convenient way to measure the strength of this relationship.

To motivate the formal theory, consider the following example. Suppose there are K assets, $i = 1, 2, \ldots, K$, observed over T periods, $t = 1, 2, \ldots, T$. Each asset has a mean return, μ_i and variance σ_i^2. Different assets have different average returns, so μ_i varies across assets i. Why would an investor hold both a high-returning asset and a low-returning asset? The simplest story might be that assets that have high returns also tend to have high variance and that if investors dislike variance, then they will be willing to hold a mix of high-return and low-return assets if the low-return assets also have low variance. This story leads to a testable implication: that there should be an upward-sloping relationship between mean and variance,

$$\mu_i = \gamma + \lambda\sigma_i^2 + u_i$$

By computing the collection of (μ_i, σ_i^2), we can test the theory by regressing the cross-section of mean returns on the cross-section of return variances. A positive value of λ indicates that as variance rises, return also rises; that is, there is higher average return as a reward for higher risk.

The cross-sectional method performs an exercise in a similar spirit. However, instead of using the variance of an asset as its measure of risk, the cross-sectional method measures risk by covariance of a return with a factor, such as market return, or economic conditions. The intuition is that assets that covary strongly with the factor are more risky. In a first step, compute the factor exposures by a time-series regression,

$$r_{it} = \alpha_i + \beta_i x_t + e_{it}$$

where the dependent variable r_{it} is the return of asset i in period t, α_i is the average return in excess of market risk, and the independent variable x_t is a common, observed factor that is proposed to be driving the asset returns. This regression is run, once per asset, to compute a cross-section of factor exposures β_i. There will be K different values for β_i, one for each asset. The value β_i for asset i is called that asset's factor exposure. A β_i of zero indicates that the asset's return does not vary with that factor and so is said to not be exposed to variation in that factor. Factor exposure greater than zero indicates that the asset in question has higher returns in periods when that factor itself is relatively high. Negative factor exposure indicates that the asset in question is a "hedge" in that its returns tend to be high precisely in periods that the returns of the factor in question are low.

In the second stage, we test whether assets that have high factor exposures also have higher average returns:

$$\mu_i = \gamma + \lambda\beta_i + u_i \tag{1}$$

In this regression, the independent variables are themselves regression coefficients from the first stage. The parameter of interest is the coefficient λ, called the price of risk, which captures by how much expected return rises with a one-unit increase in factor exposure.

The Fama–MacBeth regression

The Fama–MacBeth regression is a method for estimating the parameters of the cross-sectional model. Actually running a regression on (1) would produce correct point estimates, but standard formulas would understate the variance of $\widehat{\lambda}$ because of the regression being run on averages and the independent variables β_i being themselves estimated quantities. The Fama–MacBeth regression addresses the first problem by running (1) in each period.

Regression (1) places the $K \times 1$ vector of average returns on the left-hand side. The Fama–MacBeth regression runs T regressions, one per time period, with returns r_{it} on the left-hand side:

$$r_{it} = \gamma_t + \lambda_t\beta_i + u_i$$

In this regression, the left-hand side is the K asset returns at time t, and the right-hand side is the K first-stage estimates of β_i, one for each asset. Thus, the parameter λ_t is the slope coefficient relating the cross-section of returns at time t to the factor exposures. The parameter γ_t is the intercept term, or the zero-beta return, in time period t. Performing this regression for each t, we obtain a time series for $\{\widehat{\lambda}_t\}$. The interpretation is that $\{\lambda_t\}$ is a time series of the price of risk, period by period.

In the second step, the λ_t are averaged over time to obtain a single estimate of the price of risk over the time-period sample.

$$\hat{\lambda} = \frac{1}{T}\sum_{t=1}^{T}\hat{\lambda}_t$$

The Fama–MacBeth regression estimates the variance of $\hat{\lambda}$ via the sample variance of the $\hat{\lambda}_t$,

$$\hat{\sigma}^2_{\hat{\lambda}} = \frac{1}{T^2}\sum_{t=1}^{T}(\hat{\lambda}_t - \hat{\lambda})^2$$

The estimates of the average zero-beta excess return and its variance, $\hat{\gamma}$ and $\hat{\sigma}^2_{\hat{\gamma}}$, are obtained similarly. The method extends to multiple factors.

For a recent survey of financial regressions that includes a discussion of the Fama–MacBeth regression, see Goyal (2012, 13). For textbook treatments of financial regressions that include the Fama–MacBeth regression, see Cochrane (2005, 246) and Campbell (2018, 64).

▷ Example 1: Single-factor Fama–MacBeth regression

We use simulated data on fictional stock returns and describe the first few variables.

```
. use https://www.stata-press.com/data/r19/finex
(Fictional stock price data)
. describe datestr-wgt
```

Variable name	Storage type	Display format	Value label	Variable label
datestr	str11	%11s		String date
datem	int	%tm		Monthly date
sp500	double	%10.0g		S&P 500
vol	float	%9.0g		Volatility index
fedfunds	float	%9.0g		Federal funds rate
acme	float	%9.0g		Aciron Medical, Inc.
bat	float	%9.0g		Boron Advanced Technologies
iron	float	%9.0g		Industrial Operations Network
dune	float	%9.0g		Digital Urban Network Enterprise
tyr	float	%9.0g		Tyndale Resources Group
glo	float	%9.0g		Green Logistics, Inc.
spa	float	%9.0g		Space Rocket MFG
wgt	float	%9.0g		Widget Gadgets

This dataset contains fictional stock prices for 25 assets, monthly, from 1955 to 2019.

To prepare the data for analysis, we 1) use `finreturns` to generate monthly returns on the 25 assets; 2) use `finreturns` to generate the return on the market (which is used as a factor); and 3) transform the interest rate so that it is in the same units. The units in this example are log monthly returns.

To create log monthly returns for the 25 assets, we use `finreturns` with the `log()` option. The `multiply(100)` option scales the result so that the units are approximately percentages. A value of 0.50 indicates that the asset increased in value by 0.50 log points or about 0.5%.

```
. quietly finreturns acme-tks, log(lnr_) multiply(100)
```

The result is a set of variables `lnr_acme`, `lnr_bat`, etc, all the way up to `lnr_tks`. We treat the market index, `sp500`, in an identical manner.

```
. quietly finreturns sp500, log(lnr_mkt) multiply(100)
```

The result is a `lnr_mkt` variable that holds the log monthly return on the market. With this preprocessing complete, we can fit a Fama–MacBeth regression.

Recall that the object of interest is the relationship between the cross-section of average returns and the cross-section of time-series coefficients β. As a prelude, we run `finsummarize` to see the average log return for a few assets and their slope coefficients with respect to the market factor.

```
. finsummarize lnr_acme-lnr_tyr, benchmark(lnr_mkt) statistics(beta)
Financial summary statistics
Number of obs = 779
Sample: 1955m2 thru 2019m12
```

	Mean	Std. dev.	beta
ACME	0.4437	1.1813	0.1282
BAT	0.7534	5.4833	1.5338
IRON	0.7979	6.6804	1.8767
DUNE	0.8331	6.7105	1.8845
TYR	0.6214	3.9969	1.0878

```
Note: Variable lnr_mkt used as benchmark
      asset.
```

The `finsummarize` table shows the mean, standard deviation, and slope coefficient (`beta`) for each asset. The goal is to regress mean returns on beta to determine whether, as an asset becomes riskier (higher beta), it also generates increased return, on average, and whether there is a reward for taking on more risk, on average.

`finregress fmb` determines this by directly running that regression. We specify all 25 log return variables as dependent variables and the market factor `lnr_mkt` as the independent variable.

```
. finregress fmb lnr_acme-lnr_tks = lnr_mkt
Fama-MacBeth regression
Sample: 1955m2 thru 2019m12                          Number of obs     = 779
Method: OLS                                          Number of depvars =  25
```

depvar_means	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
beta						
lnr_mkt	.1875542	.1264035	1.48	0.138	-.060192	.4353005
_cons	.4561836	.0193535	23.57	0.000	.4182514	.4941158

```
Dependent variables: lnr_acme lnr_bat lnr_iron lnr_dune lnr_tyr lnr_glo
                     lnr_spa lnr_wgt lnr_bar lnr_yum lnr_aaa ... lnr_tks
```

The output table provides two numbers. These numbers are the coefficient on a regression of mean log returns (the dependent variable in the second stage) on beta risk (the independent variable in the second stage) and an intercept term. The slope coefficient expresses the expected rise in mean log returns as beta risk rises by one point. The coefficient on `lnr_mkt` is 0.19. This slope indicates that as beta risk rises by one unit, expected average log return rises by 0.19 points.

Although we ran the regression using log returns, the interpretation of the results is often made with respect to simple returns. For small returns, such as daily and monthly returns, we can use the slope coefficient estimated by using log returns to approximate the slope coefficient we would have obtained by using simple returns. So, in our example, we can say that as beta risk rises by one unit, we expect the average simple return to rise by 0.19 percentage points.

More generally, when we can interpret the results as simple returns, we can use the following intuition: A return on a stock with a beta of 1 will be up 1 percentage point every time the market return is also up 1 percentage point and, similarly, will be down 1 percentage point when the market return is down 1 percentage point. A return on a stock with a beta of 2 is more volatile: it will be up 2 percentage points for every time the market return is up 1 percentage point and will be down 2 percentage points whenever

the market return is down 1 percentage point. The second stock reacts twice as strongly with respect to the market return as does the first stock. The coefficient estimate of 0.19 indicates that the reward for taking on this increased volatility is a higher average return of 0.19 percentage points per month.

The second number in the coefficient table is the intercept in the regression of average log return on beta risk. Thus, it can be interpreted as the zero-beta return; that is, it is an estimate of the log return on an asset with zero beta risk (an asset that does not react to the market at all). This estimate of 0.46 means that a zero-beta asset is expected to have an average log return of 0.46 or approximately a simple return of 0.46% per month.

◁

▷ Example 2: Multifactor Fama–MacBeth regression

In the presence of independent variables, they are included in the first-stage time-series regressions. For each of the K assets, there will be M independent variables and thus M coefficients, β_{ij}, $i = 1, 2, \ldots, K$, $j = 1, 2, \ldots, M$.

The second-stage regressions become a multivariate regression of returns against the estimated first-stage coefficients, performed period by period, and the resulting single number per coefficient is the price of risk with respect to exposure to that variable.

In our dataset, we also have a fictional volatility factor, `vol`, that captures volatility each time period. Adding this factor as an independent variable to the Fama–MacBeth regression produces the following:

```
. finregress fmb lnr_acme-lnr_tks = lnr_mkt vol
Fama-MacBeth regression
Sample: 1955m2 thru 2019m12                          Number of obs     = 779
Method: OLS                                          Number of depvars =  25
```

depvar_means	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
beta						
lnr_mkt	.0239579	.1315547	0.18	0.855	-.2338845	.2818003
vol	1.384656	.3671927	3.77	0.000	.6649715	2.104341
_cons	.5841546	.046591	12.54	0.000	.4928379	.6754714

Dependent variables: **lnr_acme lnr_bat lnr_iron lnr_dune lnr_tyr lnr_glo lnr_spa lnr_wgt lnr_bar lnr_yum lnr_aaa ... lnr_tks**

The coefficient on `lnr_mkt` is essentially zero, indicating that average log returns are not expected to change as market beta risk increases. The coefficient on `vol`, the price of volatility risk, is sharply positive. This indicates that as volatility risk rises, average log returns are expected to rise. Indeed, the estimated coefficient implies that investors require an average increased log return of 1.38 for each unit of volatility risk they bear, or about 1.38% additional simple return per month. That is, assets that are highly exposed to volatility also tend to have high returns. This can be explained as compensation. If investors dislike exposure to volatility, then assets that covary highly with volatility must generate higher returns to entice investors to hold them.

◁

▷ Example 3: Dependent and independent variable adjustment

So far, we have used unadjusted returns. However, it is common to instead use returns in excess of a risk-free rate, r_t^f, as the object of interest. Thus if the dependent variable is the return on asset i, r_{it}, and the independent variables are the market return r_t^m and an additional factor x_t, then it is common to consider

$$(r_{it} - r_t^f) = \alpha + \beta_{i1}(r_t^m - r_t^f) + \beta_{i2}x_t + e_{it}$$

where the dependent variable r_{it} and some independent variables like r_t^m are expressed as excess returns.

Let's use the federal funds interest rate as a risk-free rate. It is reported at an annual rate, so we additionally adjust it to be a monthly rate:

```
. generate double rf = 100 * log(1 + fedfunds/100)/12
```

In `finregress`, the dependent variables are expressed net of a risk-free rate if the `rfrate()` option is specified. Thus, typing

```
. finregress fmb lnr_acme-lnr_tks = lnr_mkt, rfrate(rf)
```

would subtract `rf` from each dependent variable prior to fitting the first-stage time-series regressions.

To also adjust independent variables, we could specify them in the `adjust()` option. Thus, typing

```
. finregress fmb lnr_acme-lnr_tks = lnr_mkt vol, rfrate(rf) adjust(lnr_mkt)
```

would adjust both the dependent variables and the independent variable `lnr_mkt` by subtracting `rf` prior to estimation. Because `vol` does not appear in `adjust()`, no adjustment is made. Similarly, typing

```
. finregress fmb lnr_acme-lnr_tks = lnr_mkt vol, rfrate(rf) adjust(lnr_mkt vol)
```

would adjust all dependent variables and both independent variables `lnr_mkt` and `vol`. This behavior could also be obtained by typing

```
. finregress fmb lnr_acme-lnr_tks = lnr_mkt vol, rfrate(rf) adjust
```

Let's run the two-factor model above but use excess log returns for the dependent variables and for the market factor.

```
. finregress fmb lnr_acme-lnr_tks = lnr_mkt vol, rfrate(rf) adjust(lnr_mkt)
Fama-MacBeth regression
Sample: 1955m2 thru 2019m12                          Number of obs     = 779
Method: OLS                                          Number of depvars =  25
```

depvar_means	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
beta						
lnr_mkt	.1298006	.127215	1.02	0.308	-.1195363	.3791375
vol	1.383778	.3676425	3.76	0.000	.6632122	2.104344
_cons	.0907121	.0199399	4.55	0.000	.0516305	.1297936

```
Dependent variables: lnr_acme lnr_bat lnr_iron lnr_dune lnr_tyr lnr_glo
                     lnr_spa lnr_wgt lnr_bar lnr_yum lnr_aaa ... lnr_tks
Notes: Dependent variables adjusted for risk-free rate rf.
       Independent variable lnr_mkt adjusted for risk-free rate rf.
```

The result is that the market factor's price of risk rises to 0.13. The first-stage coefficient on `lnr_mkt` is now the coefficient on a regression of excess log returns on excess log market return. The associated price of the risk coefficient, shown in the table with a value of 0.13, indicates that as exposure to excess log market return rises by 1, then the excess log return on an asset is expected to rise by 0.13.

◁

Stored results

`finregress fmb` stores the following in `e()`:

Scalars

`e(N)`	number of observations
`e(k_indepvars)`	number of independent variables, not including the constant
`e(hastscons)`	`1` if a constant was included in first stage, `0` otherwise
`e(k_dv)`	number of dependent variables
`e(dfk)`	`1` if `dfk` was specified, `0` otherwise
`e(shanken)`	`1` if `shanken` was specified, `0` otherwise
`e(df_m)`	model degrees of freedom for second-stage regressions (`small` only)
`e(df_r)`	residual degrees of freedom for second-stage regressions (`small` only)
`e(tmin)`	minimum time
`e(tmax)`	maximum time

Macros

`e(cmd)`	`finregress`
`e(subcmd)`	`fmb`
`e(cmdline)`	command as typed
`e(depvar)`	names of dependent variables
`e(indepvars_unadj)`	names of unadjusted independent variables
`e(indepvars_adj)`	names of adjusted independent variables
`e(rfrate)`	name of risk-free rate variable
`e(small)`	`small`, if specified
`e(model)`	type of model
`e(title)`	title in estimation output
`e(tmaxs)`	formatted maximum time
`e(tmins)`	formatted minimum time
`e(tvar)`	time variable
`e(properties)`	`b V`
`e(estat_cmd)`	program used to implement `estat`
`e(asbalanced)`	factor variables `fvset` as `asbalanced`
`e(asobserved)`	factor variables `fvset` as `asobserved`

Matrices

`e(b)`	coefficient vector
`e(V)`	variance matrix
`e(Sigma_e)`	variance matrix of first-stage residuals
`e(beta)`	first-stage regression coefficients

Functions

`e(sample)`	marks estimation sample

In addition to the above, the following is stored in `r()`:

Matrices

`r(table)`	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in `r()` are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

Methods and formulas are presented under the following headings:

Fama–MacBeth regression

There are K returns, $i = 1, 2, \ldots, K$, and there are T time periods, $t = 1, 2, \ldots, T$. The $K \times 1$ vector of returns in period t is $\mathbf{r}_t$. There are M independent variables, and the $M \times 1$ vector of independent variables in period t is $\mathbf{x}_t$.

In the first stage, a typical time period is characterized by the model

$$\mathbf{r}_t = \mathbf{a} + \mathbf{B}\mathbf{x}_t + \mathbf{e}_t$$

where $\mathbf{a}$ is a $K \times 1$ vector of intercepts, $\mathbf{B}$ is a $K \times M$ matrix of coefficients, and $\mathbf{e}_t$ is a $K \times 1$ vector of time-series errors. The matrix $\widehat{\mathbf{B}}$ is an estimate of $\mathbf{B}$ obtained by ordinary least squares, and it contains the point estimates of the time-series regression.

In the second stage, the cross-sectional regression is run,

$$\mathbf{r}_t = \mathbf{1}_K\gamma_t + \widehat{\mathbf{B}}\boldsymbol{\lambda}_t + \mathbf{u}_t$$

where $\mathbf{1}_K$ is a vector of 1s, γ_t is an intercept term for period t, $\boldsymbol{\lambda}_t$ is an $M \times 1$ vector of coefficients for period t, and $\mathbf{u}_t$ are cross-sectional errors in period t. The cross-sectional regression is run T times, one for each t, with the result being a time series of coefficients $\{\widehat{\boldsymbol{\lambda}}_t\}_{t=1}^{T}$ and a time series of constants $\{\hat{\gamma}_t\}_{t=1}^{T}$. The final Fama–MacBeth estimates are

$$\widehat{\boldsymbol{\lambda}} = \frac{1}{T}\sum_{t=1}^{T}\widehat{\boldsymbol{\lambda}}_t \qquad\qquad \hat{\gamma} = \frac{1}{T}\sum_{t=1}^{T}\hat{\gamma}_t$$

with their corresponding variance estimates

$$\widehat{\mathbf{V}}_{\widehat{\boldsymbol{\lambda}}} = \frac{1}{T^2}\sum_{t=1}^{T}(\widehat{\boldsymbol{\lambda}}_t - \widehat{\boldsymbol{\lambda}})(\widehat{\boldsymbol{\lambda}}_t - \widehat{\boldsymbol{\lambda}})' \qquad\qquad \hat{\sigma}^2_{\hat{\gamma}} = \frac{1}{T^2}\sum_{t=1}^{T}(\hat{\gamma}_t - \hat{\gamma})^2$$

The variance calculation is based on Cochrane (2005, 246).

Details of options

When `noconstant` is specified, the first-stage (time-series) regressions are run without an intercept. An intercept is still included in the second-stage (cross-sectional) regressions.

When `gls` is specified, the cross-sectional regressions are fit with generalized least squares instead of ordinary least squares.

When `dfk` is specified, a small-sample degrees-of-freedom adjustment is applied to the second-stage regressions.

When shanken is specified, a correction is made in the second-stage regressions to account for the fact that the coefficient matrix **B** was estimated in the first stage. The Shanken correction, described in Shanken (1992) and (Goyal 2012, 14) is

$$\widetilde{\mathbf{V}}_{\widehat{\boldsymbol{\lambda}}} = \frac{1}{T}\left\{(1+c)(T\widehat{\mathbf{V}}_{\widehat{\boldsymbol{\lambda}}} - \boldsymbol{\Sigma}_{\mathbf{x}}) + \boldsymbol{\Sigma}_{\mathbf{x}}\right\}$$

where $\boldsymbol{\Sigma}_{\mathbf{x}}$ is the covariance matrix of the independent variables and $c = \widehat{\boldsymbol{\lambda}}' \boldsymbol{\Sigma}_{\mathbf{x}}^{-1} \widehat{\boldsymbol{\lambda}}$.

References

Campbell, J. Y. 2018. *Financial Decisions and Markets: A Course in Asset Pricing*. Princeton, NJ: Princeton University Press.

Campbell, J. Y., A. W. Lo, and A. C. MacKinlay. 1997. *The Econometrics of Financial Markets*. Princeton, NJ: Princeton University Press. https://doi.org/10.2307/j.ctt7skm5.

Cochrane, J. H. 2005. *Asset Pricing*. Rev. ed. Princeton, NJ: Princeton University Press.

Fama, E. F., and J. D. MacBeth. 1973. Risk, return, and equilibrium: Empirical tests. *Journal of Political Economy* 81: 607–636. https://doi.org/10.1086/260061.

Goyal, A. 2012. Empirical cross-sectional asset pricing: A survey. *Financial Markets and Portfolio Management* 26: 3–38. https://doi.org/10.1007/s11408-011-0177-7.

Shanken, J. 1992. On the estimation of beta-pricing models. *Review of Financial Studies* 5: 1–33. https://doi.org/10.1093/rfs/5.1.1.

Also see

[FIN] **finregress fmb postestimation** — Postestimation tools for finregress fmb+

[FIN] **finregress capm** — Capital asset pricing model (CAPM)+

[FIN] **finreturns** — Generate financial returns+

[U] 20 Estimation and postestimation commands

finregress fmb postestimation — Postestimation tools for finregress fmb[+]

[+]Postestimation features after `finregress fmb` are part of StataNow.

Postestimation commands

The following standard postestimation commands are available after `finregress fmb`:

Command	Description
`estat summarize`	summary statistics for the estimation sample
`estat vce`	variance–covariance matrix of the estimators (VCE)
`estimates`	cataloging estimation results
`etable`	table of estimation results
`lincom`	point estimates, standard errors, testing, and inference for linear combinations of parameters
`nlcom`	point estimates, standard errors, testing, and inference for nonlinear combinations of parameters
`test`	Wald tests of simple and composite linear hypotheses
`testnl`	Wald tests of nonlinear hypotheses

Remarks and examples

`finregress fmb` estimates the relationship between average returns and a collection of factor exposures. Afterward, estimates can be stored with `estimates` and displayed with `etable`. Tests of linear hypotheses can be performed with `test` and `lincom`. Tests of nonlinear hypotheses can be performed with `testnl` and `nlcom`.

Also see

[FIN] **finregress fmb** — Fama–MacBeth regression[+]

[U] 20 Estimation and postestimation commands

finreturns — Generate financial returns[+]

[+]This command is part of StataNow.

Description

`finreturns` takes time-series variables containing the prices of financial assets and computes various types of asset-return series, including differences, percentage differences, log differences, annualized differences, and annualized log differences. All may be adjusted with custom multiplicative and additive factors.

Quick start

Generate simple returns using asset price data in variables `p1`, `p2`, and `p3`

```
finreturns p1 p2 p3
```

Same as above, but use `r` as the stub for generated variables instead of the default stub `_simple_`

```
finreturns p1 p2 p3, simple(r)
```

Same as above, but use `s` as the stub for simple return variables and stub `log` for log return variables

```
finreturns p1 p2 p3, simple(s) log(log)
```

Generate simple returns and multiply all returns by 100 to obtain percentages

```
finreturns p1 p2 p3, simple(r) multiply(100)
```

Menu

Statistics > Financial statistics > Generate financial returns

Syntax

```
finreturns varlist [ if ] [ in ] [ , options ]
```

options	Description
Main	
`simple`[(*newvars* | *stub*[, *retopts*])]	simple returns; the default
`ratio`[(*newvars* | *stub*[, *retopts*])]	ratio returns
`fdiff`[(*newvars* | *stub*[, *retopts*])]	first-difference returns
`log`[(*newvars* | *stub*[, *retopts*])]	log returns
`annual`[(*newvars* | *stub*[, *retopts*])]	annualized ratio returns
`logannual`[(*newvars* | *stub*[, *retopts*])]	annualized log returns
`replace`	overwrite any existing variables that match *newvars* or *stub*
Options	
`multiply`(*varname* | #)	multiply generated returns by *varname* or #
`add`(*varname* | #)	adjust generated returns additively by *varname* or #
`annualfreq`(#)	annualize `annual()` and `logannual()` returns by #
`ceiling`	round default annualization frequency with function `ceil()` instead of with function `floor()`
`preview`(*listspec*)	customize preview of generated returns
`nopreview`	suppress preview of generated returns
`numbered`	index new variable names with numbers instead of asset price variable names
`stlabel`	use verbose Stata variable labels for generated variables
`float`	set type of generated variables to `float`

You must `tsset` your data before using `finreturns`; see [TS] **tsset**.

retopts	Description
`multiply`(*varname* | #)	multiply generated returns by *varname* or #
`add`(*varname* | #)	adjust generated returns additively by *varname* or #
`numbered`	index new variable names with numbers instead of asset price variable names

listspec is

```
[ varlist ] [ in ] [ , listopts ]
```

where *listopts* may be any option for `list`, except for the options listed as summary options and advanced options in [D] **list**.

Options

Main

`simple`[(*newvars* | *stub*[, *retopts*])] computes simple returns, which are the proportion change in prices from the previous period. Thus, simple returns are calculated as the difference between the current and lagged price, divided by lagged price. These are often called net returns. You may specify `simple` with no arguments to assign default Stata variable names to the generated returns. The default names are the names of the specified asset price variables prefixed with `_simple_`. The argument may be a list of new variable names for the generated returns. Or the argument may be a stub, in which case the generated returns will be named *stubvar1*, *stubvar2*, etc., where *var1*, *var2*, and so on are the names of the specified asset price variables, or *stub*1, *stub*2, and so on if the `numbered` option is specified. When returns are computed for a single variable, such as `finreturns p1`, the name provided in `simple()` is used as the variable name. When returns are computed for multiple variables, such as `finreturns p1 p2 p3`, the name provided in `simple()` is assumed to be a stub.

retopts can be `multiply()` or `add()` and `numbered`; see their descriptions below.

`ratio`[(*newvars* | *stub*[, *retopts*])] computes the ratio of the current price to the lagged price. These are often called gross returns. You may specify `ratio` with no arguments to assign default Stata variable names to the generated returns. The default names are the names of the specified asset price variables prefixed with `_ratio_`. The argument may be a list of new variable names for the generated returns. Or the argument may be a stub, in which case the generated returns will be named *stubvar1*, *stubvar2*, etc., where *var1*, *var2*, and so on are the names of the specified asset price variables, or *stub*1, *stub*2, and so on if the `numbered` option is specified. When returns are computed for a single variable, such as `finreturns p1`, the name provided in `ratio()` is used as the variable name. When returns are computed for multiple variables, such as `finreturns p1 p2 p3`, the name provided in `ratio()` is assumed to be a stub.

retopts can be `multiply()` or `add()` and `numbered`; see their descriptions below.

`fdiff`[(*newvars* | *stub*[, *retopts*])] computes the first difference of prices. You may specify `fdiff` with no arguments to assign default Stata variable names to the generated returns. The default names are the names of the specified asset price variables prefixed with `_fdiff_`. The argument may be a list of new variable names for the generated returns. Or the argument may be a stub, in which case the generated returns will be named *stubvar1*, *stubvar2*, etc., where *var1*, *var2*, and so on are the names of the specified asset price variables, or *stub*1, *stub*2, and so on if the `numbered` option is specified. When returns are computed for a single variable, such as `finreturns p1`, the name provided in `fdiff()` is used as the variable name. When returns are computed for multiple variables, such as `finreturns p1 p2 p3`, the name provided in `fdiff()` is assumed to be a stub.

retopts can be `multiply()` or `add()` and `numbered`; see their descriptions below.

`log`[(*newvars* | *stub*[, *retopts*])] computes log returns, which are the logarithm of the price in the current period minus the logarithm of the price in the previous period. When returns are small, log returns approximate simple returns. You may specify `log` with no arguments to assign default Stata variable names to the generated returns. The default names are the names of the specified asset price variables prefixed with `_log_`. The argument may be a list of new variable names for the generated returns. Or the argument may be a stub, in which case the generated returns will be named *stubvar1*, *stubvar2*, etc., where *var1*, *var2*, and so on are the names of the specified asset price variables, or *stub*1, *stub*2, and so on if the `numbered` option is specified. When returns are computed for a single variable, such

as finreturns p1, the name provided in log() is used as the variable name. When returns are computed for multiple variables, such as finreturns p1 p2 p3, the name provided in log() is assumed to be a stub.

retopts can be multiply() or add() and numbered; see their descriptions below.

annual[(*newvars* | *stub*[, *retopts*])] computes the annual return, which is the ratio return at an annual rate. You may specify annual with no arguments to assign default Stata variable names to the generated returns. The default names are the names of the specified asset price variables prefixed with _annual_. The argument may be a list of new variable names for the generated returns. Or the argument may be a stub, in which case the generated returns will be named *stubvar1*, *stubvar2*, etc., where *var1*, *var2*, and so on are the names of the specified asset price variables, or *stub*1, *stub*2, and so on if the numbered option is specified. When returns are computed for a single variable, such as finreturns p1, the name provided in annual() is used as the variable name. When returns are computed for multiple variables, such as finreturns p1 p2 p3, the name provided in annual() is assumed to be a stub.

retopts can be multiply() or add() and numbered; see their descriptions below.

logannual[(*newvars* | *stub*[, *retopts*])] computes the log return at an annual rate. You may specify logannual with no arguments to assign default Stata variable names to the generated returns. The default names are the names of the specified asset price variables prefixed with _logannual_. The argument may be a list of new variable names for the generated returns. Or the argument may be a stub, in which case the generated returns will be named *stubvar1*, *stubvar2*, etc., where *var1*, *var2*, and so on are the names of the specified asset price variables, or *stub*1, *stub*2, and so on if the numbered option is specified. When returns are computed for a single variable, such as finreturns p1, the name provided in logannual() is used as the variable name. When returns are computed for multiple variables, such as finreturns p1 p2 p3, the name provided in logannual() is assumed to be a stub.

retopts can be multiply() or add() and numbered; see their descriptions below.

replace specifies that any existing variables having the same names specified in *newvars* or *stub* may be overwritten.

Options

multiply(*varname* | #) multiplies all generated asset returns by the variable *varname* or by the constant #. Adjusting by *varname* allows observation-by-observation adjustment rather than the adjustment applied by a constant, multiply(#). Most usefully, multiply(100) transforms the generated asset returns from proportions (such as 0.015) to percentages (such as 1.5). Only one of multiply() or add() may be specified.

When multiply() is specified as part of *retopts*, it affects only the returns generated within the corresponding option such as simple() or log(). You cannot specify multiply() both globally and within the options.

add(*varname* | #) adds *varname* or # to all generated asset returns. add(*varname*) adds the values in *varname* to the generated asset returns; this allows observation-by-observation adjustment. add(#) adds the constant # to the generated asset returns. For example, with this option, a variable such as inflation can be added to or subtracted from all generated returns, so that the generated returns are real returns. Only one of add() or multiply() may be specified.

When add() is specified as part of *retopts*, it affects only the returns generated within the corresponding option such as simple() or log(). You cannot specify add() both globally and within the options.

annualfreq(#) specifies the integer factor used to annualize returns created by annual and logannual. By default, the scaling implied by the frequency of the data is used: 365 for daily data, 12 for monthly data, and 4 for quarterly data. If the data are tsset with a delta other than 1, these default frequencies are 365/delta, 12/delta, and 4/delta, respectively, rounded with the floor() function by default or with the ceil() function if the ceiling option is specified.

ceiling specifies that the default annualization frequency be rounded up with the ceil() function rather than rounded down with the floor() function. Custom annualization frequencies specified with annualfreq() are not rounded, so ceiling is ignored when annualfreq() is specified.

preview(*listspec*) customizes the preview, specifying which variables and observations to list and controlling the format of lists.

nopreview specifies that the preview of return variables should be suppressed from the output.

numbered specifies that new return variable names created by default or by specifying a stub be numbered. The default is to use the asset price variable name in the new return variable name. For example, if the asset price variables are aaa and bbb, then the default simple returns are named _simple_aaa and _simple_bbb. When numbered is specified, the return variables are named _simple_1 and _simple_2. Similarly, if simple(s_) is specified to indicate that s_ be used as a stub, then new variables are named s_aaa and s_bbb by default but named s_1 and s_2 when numbered is specified.

When numbered is specified as part of *retopts*, it affects only the returns generated within the corresponding option such as simple() or log().

stlabel labels the generated returns with verbose labels. By default, the variable label is the capitalized form of the specified asset price variable.

float specifies that the type of the generated variables be float. By default, all return variables are generated as double regardless of the storage type set by users.

Remarks and examples

finreturns generates growth rate variables representing returns from an existing list of variables representing prices. Several growth rates are available, with different properties that may be more or less appropriate for different tasks.

The first difference, sometimes called the dollar return, is just the change in price between two periods,

$$P_t - P_{t-1}$$

which provides a measure of absolute change. More commonly used is a relative or proportional change. This can be in gross terms, known as the ratio return,

$$\frac{P_t}{P_{t-1}}$$

or in net terms, which is known as the simple return:

$$\frac{P_t}{P_{t-1}} - 1$$

Unlike first differences, these are scale-free measures.

However, relative returns are not additive. That is, a 1% increase in a stock's price, followed by a 1% decrease, does not yield back the original price. This is because $(1+0.01)(1-0.01) \neq 1$. For example, if a stock's price falls by 50%, then it must rise by 100% to return to the original value. In some statistical analyses of asset returns, it is important to consider sums of returns. In these situations, the log return is often used:

$$\log P_t - \log P_{t-1} = \log\left(\frac{P_t}{P_{t-1}}\right)$$

For small returns, the log return is close to the simple return, but log returns are additive; a log return of 0.01 in P_t, followed by a log return of -0.01, corresponds to a zero net change in P_t.

Simple and log returns are measured in proportions, so a value of, say, 0.01 corresponds to a 0.01 proportional increase in the price of an asset. Analysts sometimes wish to scale their returns to percentages,

$$100 \times \left(\frac{P_t}{P_{t-1}} - 1\right)$$
$$100 \times (\log P_t - \log P_{t-1})$$

Other times, we may wish to adjust for a time-varying variable. For example, if inflation from time $t-1$ to time t is denoted π_t, then real returns may be expressed as

$$r_t^{\text{gross,real}} = \frac{P_t}{P_{t-1}} \times \frac{1}{1+\pi_t}$$
$$r_t^{\text{log,real}} = \log P_t - \log P_{t-1} - \pi_t$$

where we use the fact that $\log(1+\pi_t)$ is approximately π_t for small values of π_t. The adjustment factors here vary over time, rather than being a constant. `finreturns` allows either multiplicative or additive adjustments, though not both at the same time.

The returns we have discussed so far are one-period returns. But sometimes we wish to express returns at an annual rate. This corresponds to the situation that a monthly return continues for 12 months, a quarterly return for 4 quarters, and so on. `finreturns` can compute annualized ratio returns,

$$\left(\frac{P_t}{P_{t-1}}\right)^m - 1$$

where m is the compounding factor or annualization frequency equal to 12 for monthly data, 4 for quarterly data, etc., and log annual-returns,

$$\log\left(\frac{P_t}{P_{t-1}}\right) \times m$$

Example 1: Simple returns

We use a dataset with stock prices on 25 fictional companies and describe the first few variables.

```
. use https://www.stata-press.com/data/r19/finex
(Fictional stock price data)
. describe datestr-wgt
```

Variable name	Storage type	Display format	Value label	Variable label
datestr	str11	%11s		String date
datem	int	%tm		Monthly date
sp500	double	%10.0g		S&P 500
vol	float	%9.0g		Volatility index
fedfunds	float	%9.0g		Federal funds rate
acme	float	%9.0g		Aciron Medical, Inc.
bat	float	%9.0g		Boron Advanced Technologies
iron	float	%9.0g		Industrial Operations Network
dune	float	%9.0g		Digital Urban Network Enterprise
tyr	float	%9.0g		Tyndale Resources Group
glo	float	%9.0g		Green Logistics, Inc.
spa	float	%9.0g		Space Rocket MFG
wgt	float	%9.0g		Widget Gadgets

We first create the simple return on stock `acme`.

```
. finreturns acme
Simple returns generated for variable acme:
```

	datem	acme	_simple_acme
1.	1955m1	10.00609	.
2.	1955m2	10.24331	.02370763
3.	1955m3	10.38311	.01364765
4.	1955m4	10.49098	.01038863
5.	1955m5	10.59686	.0100931
6.	1955m6	10.75985	.01538048
7.	1955m7	10.67278	-.00809163
8.	1955m8	10.63304	-.00372363
9.	1955m9	10.59028	-.00402204
10.	1955m10	10.48927	-.00953793

By default, the simple return is computed. Because no information about the new variable name was provided, `finreturns` generated a variable with the name `_simple_acme`. The value of 0.0237 in observation 2 of the new variable corresponds to the simple return from period 1 to period 2, an increase of 2.37%.

The output consists of a listing of the first 10 observations of the original variable (`acme`) and all variables generated from that original variable.

When multiple stocks are specified, `finreturns` produces returns for each of them.

```
. finreturns bat iron
Simple returns generated for variable bat:
```

	datem	bat	_simple_bat
1.	1955m1	10.00766	.
2.	1955m2	10.63947	.06313206
3.	1955m3	10.37636	-.02472954
4.	1955m4	10.87464	.04802038
5.	1955m5	10.76794	-.00981138
6.	1955m6	11.72315	.08870823
7.	1955m7	13.10283	.11768852
8.	1955m8	12.848	-.01944843
9.	1955m9	13.79885	.07400814
10.	1955m10	12.84272	-.06929062

```
Simple returns generated for variable iron:
```

	datem	iron	_simple_iron
1.	1955m1	10.00591	.
2.	1955m2	10.45016	.0443978
3.	1955m3	10.29193	-.01514066
4.	1955m4	10.83185	.05246058
5.	1955m5	10.77944	-.00483852
6.	1955m6	11.84796	.09912496
7.	1955m7	13.68152	.15475758
8.	1955m8	13.35943	-.02354157
9.	1955m9	14.41827	.07925792
10.	1955m10	13.08083	-.09276043

Using the `preview()` option, we could customize the look of this output.

If we `describe` the returns generated so far, we will find that `finreturns` has given the generated variables some labels.

```
. describe _simple*
Variable      Storage   Display    Value
    name         type    format    label      Variable label
------------------------------------------------------------------------------
_simple_acme    double  %10.0g                ACME
_simple_bat     double  %10.0g                BAT
_simple_iron    double  %10.0g                IRON
```

The default label is the capitalized version of the original variable name. If the original variable names are stock tickers, as will often be the case in financial applications, then the labels on the returns will also be the tickers. The `stlabel` option provides a more verbose label.

Often, you will wish to create many returns series at once and will not wish to type all the price series' variable names. The `finreturns` command understands Stata's varlist conventions; see **[U] 11.4 varname and varlists**. In particular, the notation *varname1-varname2* means all variables between the two specified variable names, inclusive. This feature can be combined with the ability to specify a stub for the returns variable names in `simple()`. We type the `finreturns` command and then use `describe` to see the new variables:

```
. finreturns acme-wgt, simple(r_)
  (output omitted)
. describe r*
```

Variable name	Storage type	Display format	Value label	Variable label
r_acme	double	%10.0g		ACME
r_bat	double	%10.0g		BAT
r_iron	double	%10.0g		IRON
r_dune	double	%10.0g		DUNE
r_tyr	double	%10.0g		TYR
r_glo	double	%10.0g		GLO
r_spa	double	%10.0g		SPA
r_wgt	double	%10.0g		WGT

Eight new returns series—`r_acme`, `r_bat`, and so on—were created, each corresponding to one variable in the original list. The labels on the returns inform you to which series the return corresponds.

◁

▷ Example 2: Different types of returns

So far, we have looked only at simple returns. But `finreturns` can produce multiple types of returns simultaneously. Let's use one stock price `epg` and the `ratio()`, `simple()`, and `log()` options, which each create one new variable, to illustrate generating multiple return types. We also use the `stlabel` option to use more descriptive variable labels. We then use `describe` on the generated variables to see that each one is labeled with more verbose variable labels.

```
. finreturns epg, ratio(ratio_epg) simple(net_epg) log(log_epg) stlabel
Ratio, simple, and log returns generated for variable epg:
```

	datem	epg	ratio_epg	net_epg	log_epg
1.	1955m1	10.00759	.	.	.
2.	1955m2	10.24591	1.023814	.02381396	.02353483
3.	1955m3	10.17058	.99264819	-.00735181	-.00737897
4.	1955m4	10.2799	1.0107486	.01074861	.01069126
5.	1955m5	10.39898	1.011583	.01158298	.01151642
6.	1955m6	10.9156	1.0496805	.04968051	.04848584
7.	1955m7	11.94555	1.0943553	.09435535	.09016546
8.	1955m8	11.92886	.99860297	-.00139703	-.00139801
9.	1955m9	12.75635	1.0693691	.06936906	.06706881
10.	1955m10	12.04943	.94458278	-.05541722	-.05701195

```
. describe *_epg
Variable      Storage   Display    Value
    name         type    format    label      Variable label
------------------------------------------------------------------------------
ratio_epg       double  %10.0g                Ratio returns for epg
net_epg         double  %10.0g                Simple returns for epg
log_epg         double  %10.0g                Log returns for epg
```

The preview displays the first 10 observations on each variable we created, as well as the original variable. The ratio (gross) return is one plus the simple (net) return. The log return is the log difference; because the log differences are fairly small, the log returns are close to the simple returns.

◁

▷ Example 3: Adjusting the returns

The returns are stored as proportions, log differences, annualized percentages, or annualized log differences. Some analysts find it easier to interpret percentages, such as 10 rather than 0.1. The `multiply()` option makes converting to percentages easy.

```
. finreturns das, simple(r_das) multiply(100)
(all returns multiplied by 100)
Simple returns generated for variable das:
```

	datem	das	r_das
1.	1955m1	10.00926	.
2.	1955m2	10.31771	3.0816631
3.	1955m3	10.30525	-.12077009
4.	1955m4	10.35527	.48538577
5.	1955m5	10.31676	-.37190946
6.	1955m6	10.37442	.55891598
7.	1955m7	10.77979	3.9074243
8.	1955m8	10.8401	.55941417
9.	1955m9	11.06466	2.0715751
10.	1955m10	10.59127	-4.278394

The returns are now in percentages, still at a monthly rate because the underlying data are monthly.

Sometimes, the variable we would like to adjust by is not a constant. For example, consider the notion of excess returns, which is the return on one asset minus the return on a reference asset. The `add()` option makes computation of such objects easy. Let's find the return of stock `khc` in excess of the return of a reference asset, the nominal interest rate in the `fedfunds` variable.

We first create the variable we wish to adjust by. It is the negative of the interest rate, in monthly percentage changes:

```
. generate double ffr = -fedfunds/1200
```

We divide the interest rate by 100 to get proportions and divide by 12 to get a monthly rate. We also invert the sign, making it negative, because `add()` is additive, not subtractive. Then we can call `finreturns`.

```
. finreturns khc, simple(khc1) add(ffr) preview(in 1/5)
(all returns adjusted additively by ffr)
Simple returns generated for variable khc:
```

	datem	khc	khc1
1.	1955m1	10.00526	.
2.	1955m2	10.34669	.03305012
3.	1955m3	10.2937	-.00624616
4.	1955m4	10.44079	.01309755
5.	1955m5	10.66776	.02054699

In observation 2, stock `khc` returned 3.3% over the monthly interest rate.

◁

▷ Example 4: Annualization of daily returns

To annualize means to calculate the annual equivalent of returns that were measured at a higher frequency, m. Monthly returns, for instance, are measured at a frequency of $m = 12$ times a year. Annual returns are calculated as

$$r_t^{\text{annual}} = \left(1 + r_t^{\text{simple}}\right)^m - 1$$

where $r_t^{\text{simple}} = (P_t/P_{t-1}) - 1$.

But what is the correct m to annualize a daily return? This depends on the number of days a return is actually generated each year, which in turn depends on whether the asset is traded every day of the year and on whether a year is common or leap. In the context of stocks, a popular approach is to assume 20 trading days per month (that is, approximately 4 weeks minus their weekends), which translates to an m of 20 days $\times$ 12 months $= 240$.

To annualize the returns of the `afh` stock using this rule of thumb, we type

```
. finreturns afh, annual(afh_yr240) annualfreq(240)
(annualization frequency m = 240)
Annual returns generated for variable afh:
```

	datem	afh	afh_yr240
1.	1955m1	10.0088	.
2.	1955m2	10.03054	.68326427
3.	1955m3	10.24619	163.85376
4.	1955m4	10.1483	-.90012114
5.	1955m5	10.49291	3022.9268
6.	1955m6	10.62643	19.796663
7.	1955m7	11.34264	6287254.9
8.	1955m8	11.20041	-.95161606
9.	1955m9	11.60558	5055.8637
10.	1955m10	11.05977	-.99999047

Note that, if you do not specify a frequency with the annualfreq() option, finreturns assumes 365 trading days per year, which may or may not be appropriate for the assets you want to study.

◁

Stored results

finreturns stores the following in r():

Scalars

r(annualfreq)	frequency for annualization
r(ceiling)	1 if ceiling specified, 0 otherwise

Macros

r(varlist)	variable names in *varlist*
r(newvars)	names of all variables created by finreturns
r(simplevars)	new variables storing simple returns, if specified
r(ratiovars)	new variables storing ratio returns, if specified
r(fdiffvars)	new variables storing first-difference returns, if specified
r(logvars)	new variables storing log returns, if specified
r(annualvars)	new variables storing annualized ratio returns, if specified
r(logannualvars)	new variables storing annualized log returns, if specified
r(multiply)	constant or variable name used for multiplicative adjustment
r(add)	constant or variable name used for additive adjustment
r(simple_multiply)	multiplicative adjustment used for simple returns
r(ratio_multiply)	multiplicative adjustment used for ratio returns
r(fdiff_multiply)	multiplicative adjustment used for first-difference returns
r(log_multiply)	multiplicative adjustment used for log returns
r(annual_multiply)	multiplicative adjustment used for annualized ratio returns
r(logannual_multiply)	multiplicative adjustment used for annualized log returns
r(simple_add)	additive adjustment used for simple returns
r(ratio_add)	additive adjustment used for ratio returns
r(fdiff_add)	additive adjustment used for first-difference returns
r(log_add)	additive adjustment used for log returns
r(annual_add)	additive adjustment used for annualized ratio returns
r(logannual_add)	additive adjustment used for annualized log returns

Methods and formulas

Methods and formulas are presented under the following headings:

Basic formulas

Let P_t be any time series, though we will usually think of it as a price series. Then the returns generated by the ratio() option are calculated as

$$r_t^{\text{ratio}} = \frac{P_t}{P_{t-1}}$$

The returns generated by the fdiff() option are calculated as

$$r_t^{\text{fdiff}} = P_t - P_{t-1}$$

The returns generated by the `simple()` option are calculated as

$$r_t^{\text{simple}} = \frac{P_t - P_{t-1}}{P_{t-1}} = \frac{P_t}{P_{t-1}} - 1 = r_t^{\text{ratio}} - 1$$

The returns generated by the `log()` option are calculated as

$$r_t^{\log} = \log\left(\frac{P_t}{P_{t-1}}\right) = \log P_t - \log P_{t-1}$$

The returns generated by the `annual()` option are calculated as

$$r_t^{\text{annual}} = \left(r_t^{\text{ratio}}\right)^m - 1$$

where m depends on the frequency of the data. For quarterly data, $m = 4$; for monthly data, $m = 12$; for weekly data, $m = 52$; and for daily data, $m = 365$.

The returns generated by the `logannual()` option are calculated as

$$\log\left(\frac{P_t}{P_{t-1}}\right) \times m$$

Options for adjusting returns

The `multiply()` and `add()` options adjust the generated returns.

`multiply(#)` multiplies all generated returns by a constant. For instance, `multiply(100)` multiplies all returns by 100, thus converting proportions into percentages.

`multiply(`*varname*`)` multiplies each observation in the return series by the corresponding value in the specified *varname*. If that variable takes on value c_t at time t, then this option computes

$$r_t^{\text{adj}} = r_t c_t$$

The `add()` option makes the adjustment additively rather than multiplicatively and can also be specified with either a constant or a *varname*. Hence,

$$r_t^{\text{adj}} = r_t + c_t$$

For example, if the returns were nominal, and the variable c_t held the negative of the inflation rate, then `add()` would provide a one-line solution for obtaining real returns.

References

Campbell, J. Y. 2018. *Financial Decisions and Markets: A Course in Asset Pricing*. Princeton, NJ: Princeton University Press.

Hurn, S., V. L. Martin, P. C. B. Phillips, and J. Yu. 2020. *Financial Econometric Modeling*. Oxford: Oxford University Press.

Also see

[FIN] **finportfolio** — Financial portfolio selection+

[TS] **tsset** — Declare data to be time-series data

finsummarize — Financial summary statistics[+]

[+]This command is part of StataNow.

Description

`finsummarize` calculates six summary statistics for financial assets or portfolios: the mean, standard deviation, Sharpe ratio, Treynor index, Jensen's alpha, and the market beta. When a benchmark asset return is specified, `finsummarize` provides all statistics. Otherwise, the mean, standard deviation, and Sharpe ratio are provided.

Quick start

Summarize portfolio `myport` to obtain its mean, standard deviation, and Sharpe ratio

```
finsummarize myport
```

Same as above, but also use `mktrisk` as the return of the benchmark asset and the risk-free rate variable `rf`, and calculate the Treynor index, Jensen's alpha, and market beta

```
finsummarize myport, rfrate(rf) benchmark(mktrisk)
```

Same as above, but request only the Treynor index statistic

```
finsummarize myport, rfrate(rf) benchmark(mktrisk) statistics(treynor)
```

Menu

Statistics > Financial statistics > Financial summary statistics

Syntax

```
finsummarize varlist [ if ] [ in ] [ weight ] [ , options ]
```

options	Description
Main	
rfrate(*varname*)	specify variable containing the risk-free rate
benchmark(*varname*[, adjust])	specify the benchmark asset return variable
statistics(*stats*)	specify a list of statistics; default is all available statistics
detail	request detailed summary statistics
* nodfk	do not adjust variance estimate for degrees of freedom
* post	post the results from option detail in e()
SE/Robust	
* vce(*vcetype*)	*vcetype* may be robust, cluster *clustvar*, or hac *hacspec*
Reporting	
novarlabel	display variable names rather than variable labels
* coeflegend	display legend instead of statistics

You must tsset your data before using finsummarize; see [TS] **tsset**.

varlist and *varname* may contain time-series operators; see **[U] 11.4.4 Time-series varlists**.

collect is allowed; see **[U] 11.1.10 Prefix commands**.

aweights and fweights are allowed; see **[U] 11.1.6 weight**.

* nodfk, post, vce(), and coeflegend are meaningful only when the detail option is specified.

coeflegend does not appear in the dialog box.

stats	Description
sharpe	Sharpe ratio
treynor	Treynor index
jensen	Jensen's alpha
beta	market beta

Options

Main

`rfrate(`*varname*`)` specifies *varname* as the risk-free rate for computation of the Sharpe ratio, Treynor index, Jensen's alpha, and the market beta. If no variable is specified, `finsummarize` assumes the risk-free rate is equal to zero.

`benchmark(`*varname*`[ , adjust ])` specifies *varname* as the benchmark asset return. A benchmark asset is required to compute the Treynor index, Jensen's alpha, and the market beta. If `benchmark()` is not specified, `finsummarize` will default to reporting only the mean, standard deviation, and Sharpe ratio. `adjust` specifies that excess returns of *varname* be used in computations (that is, returns in excess of the risk-free rate specified in `rfrate()`).

`statistics(`*stats*`)` specifies a list of statistics to be computed. *stats* may include `sharpe`, `treynor`, `jensen`, and `beta`. If `statistics()` is not specified, `finsummarize` reports all possible statistics.

`detail` produces standard error estimates in addition to point estimates. All asset summary statistics are estimated jointly, allowing for tests across performance measures and across assets.

`nodfk` specifies that a small-sample degrees-of-freedom adjustment not be made when estimating the variance of the returns. By default, a small-sample degrees-of-freedom adjustment is made. `nodfk` is applicable only when the `detail` option is specified.

`post` posts the results of the underlying regression run by the `detail` option as estimation results in `e()`.

SE/Robust

`vce(`*vcetype*`)` specifies the type of standard error reported, which includes types that are robust to some kinds of misspecification (`robust`) and that allow for intragroup correlation (`cluster` *clustvar*); see [R] ***vce_option***. `vce()` is applicable only when the `detail` option is specified.

`vce(hac` *hacspec*`)` requests a heteroskedasticity- and autocorrelation-consistent (HAC) variance–covariance matrix. The full syntax of *hacspec* is one of the following:

`vce(hac` *kernel* `[` #`])` requests a HAC variance–covariance matrix using the specified kernel (see below) with optional # lags. The bandwidth of a kernel is equal to # + 1. If # is not specified, a kernel with $N - 2$ lags is used, where N is the sample size.

`vce(hac` *kernel* `opt [` #`])` requests a HAC variance–covariance matrix using the specified kernel (see below), and the lag order is selected using Newey and West's (1994) optimal lag-selection algorithm. # is an optional tuning parameter that affects the lag order selected; see the discussion in *Methods and formulas* in [R] **ivregress**.

kernel may be one of the following:

`bartlett` or `nwest` requests the Bartlett (Newey–West) kernel.

`parzen` or `gallant` requests the Parzen (Gallant 1987) kernel.

`quadraticspectral` or `andrews` requests the quadratic spectral (Andrews 1991) kernel.

Reporting

`novarlabel` specifies that the variable names be displayed instead of the variable labels. By default, variable labels are used to label assets in the output. Note that the variable names are displayed when the `detail` option is specified.

The following option is available with `finsummarize` but is not shown in the dialog box:

`coeflegend`; see [R] **Estimation options**. `coeflegend` is applicable only when the `detail` option is specified.

Remarks and examples

Financial assets and portfolios are summarized by the mean and variance of their returns, as well as simple functions of the mean and variance. `finsummarize` produces tables of these statistics.

The simplest characteristics of an asset or a portfolio's performance are its mean and standard deviation; the former captures average return and the latter captures return variability.

The Sharpe ratio is the return-to-risk ratio. If asset or portfolio i has mean return μ_i with standard deviation σ_i, and there exists a risk-free return r^f, then the Sharpe ratio is

$$S = \frac{\mu_i - r^f}{\sigma_i}$$

The numerator of this expression is the excess return of the asset. The denominator of this expression is the variability of the asset. Taken together, the Sharpe ratio expresses excess return in units of variability. A Sharpe ratio of zero indicates that the asset under consideration does not provide any excess return over the risk-free asset. A Sharpe ratio of one indicates that for every additional percentage point of variability, the asset returns one additional percentage point of return over the risk-free asset.

For a time-varying risk-free rate r_t^f, the Sharpe ratio is the average excess return divided by excess standard deviation. Thus, the Sharpe ratio becomes

$$S = \frac{\mu_i - \mu_r^f}{\sigma_e}$$

where μ_r^f is the average risk-free rate (so that $\mu_i - \mu_r^f$ is the average excess return) and σ_e is the standard deviation of excess returns, which are defined as $r_t - r_t^f$.

The market beta statistic and Jensen's alpha are closely related, because they are both parameters of a regression of excess returns to a given asset on excess returns to a benchmark asset. If asset i has return r_{it} in period t, the benchmark asset has return r_t^m, and the risk-free rate is r_t^f, then the regression

$$r_{it} - r_t^f = \alpha_i + \beta_i(r_t^m - r_t^f) + e_{it}$$

yields estimates of β_i and α_i. The statistic β_i, called the market beta, captures covariance of the asset with the benchmark, often termed systematic risk. The statistic α_i, called Jensen's alpha, measures average return in excess of market risk.

The formula above assumes that we have adjusted for the benchmark asset return by specifying the `adjust` suboption of the `benchmark()` option. For example, it is assumed that we typed

```
. finsummarize myport, rfrate(rf) benchmark(mktrisk, adjust)
```

However, without `adjust`, the regression above becomes

$$r_{it} - r_t^f = \alpha_i + \beta_i r_t^m + e_{it}$$

The Treynor index is similar to the Sharpe ratio in that it is a ratio of excess return to risk, but it uses β_i as its measure of risk:

$$T = \frac{\mu_i - r^f}{\beta_i}$$

Whereas the Sharpe ratio penalizes assets for their variability (σ_i), the Treynor index penalizes assets only for their covariance with the benchmark asset (β_i). According to the Treynor index, assets with high β_i are considered to be riskier.

▷ Example 1: Asset characteristics

We have data on 25 fictional companies, along with true data on the S&P 500 index and the federal funds rate. We describe the first few variables.

```
. use https://www.stata-press.com/data/r19/finex
(Fictional stock price data)
. describe datestr-wgt
```

Variable name	Storage type	Display format	Value label	Variable label
datestr	str11	%11s		String date
datem	int	%tm		Monthly date
sp500	double	%10.0g		S&P 500
vol	float	%9.0g		Volatility index
fedfunds	float	%9.0g		Federal funds rate
acme	float	%9.0g		Aciron Medical, Inc.
bat	float	%9.0g		Boron Advanced Technologies
iron	float	%9.0g		Industrial Operations Network
dune	float	%9.0g		Digital Urban Network Enterprise
tyr	float	%9.0g		Tyndale Resources Group
glo	float	%9.0g		Green Logistics, Inc.
spa	float	%9.0g		Space Rocket MFG
wgt	float	%9.0g		Widget Gadgets

Our dataset records prices, but portfolios are constructed on the basis of returns. We use `finreturns` to generate 25 returns series.

```
. quietly finreturns acme-tks, simple(r_) multiply(100)
```

The `simple(r_)` option in `finreturns` created simple, one-period returns for each of the assets in the variable list. The resulting returns are stored in variables named `r_acme` through `r_tks`. The `multiply(100)` option multiplied the resulting returns by 100, so that they can be interpreted as percentage changes.

From here, we can use `finsummarize` to display summary statistics for the returns of each asset.

```
. finsummarize r_acme-r_tyr
Financial summary statistics
Number of obs = 779
Sample: 1955m2 thru 2019m12
```

	Mean	Std. dev.	Sharpe ratio
ACME	0.4517	1.1856	0.3810
BAT	0.9052	5.4073	0.1674
IRON	1.0214	6.5625	0.1556
DUNE	1.0593	6.6088	0.1603
TYR	0.7028	3.9642	0.1773

The header provides information about the number of observations and the sample range. The data are monthly, so we have 779 observations from the second month of 1955 through the final month of 2019.

Because the data are monthly and we requested simple returns, the `Mean` column displays the average monthly return on the selected assets. The `Std. dev.` column displays the standard deviation of the returns. Because we did not specify a risk-free asset, the implied risk-free rate is 0 in every period, so the Sharpe ratio is just the ratio of the mean of an asset's return to the standard deviation of that asset's return.

◁

▷ Example 2: Adding a risk-free rate and benchmark return

Next we add a risk-free rate and a benchmark asset return to the computation of summary statistics. First, we divide the annual federal funds rate by 12 to get a monthly risk-free interest rate.

```
. generate double rf = fedfunds / 12
```

Next, we use `finreturns` to compute the monthly simple return on our benchmark asset (`sp500`).

```
. quietly finreturns sp500, simple(rmkt) multiply(100)
```

The above `finreturns` command creates a new variable, `rmkt`, with the simple monthly return on `sp500`, and, as before, the result is multiplied by 100 so that the resulting return can be interpreted in monthly percentage change.

With this setup complete, we compute summary statistics for our five assets:

```
. finsummarize r_acme-r_tyr, rfrate(rf) benchmark(rmkt, adjust)
Financial summary statistics
Number of obs = 779
Sample: 1955m2 thru 2019m12
```

	Mean	Std. dev.	Sharpe ratio	Treynor index	Jensen's alpha	beta
ACME	0.4517	1.1856	0.0443	0.3816	0.0192	0.1340
BAT	0.9052	5.4073	0.0927	0.3316	0.1421	1.5217
IRON	1.0214	6.5625	0.0941	0.3351	0.1794	1.8526
DUNE	1.0593	6.6088	0.0991	0.3534	0.2147	1.8640
TYR	0.7028	3.9642	0.0756	0.2780	0.0433	1.0870

```
Notes: Variables adjusted for risk-free rate rf.
       Variable rmkt used as benchmark asset and adjusted for risk-free rate
       rf.
```

By default, `finsummarize` computes all the statistics it is able to, given the information supplied in the `rfrate()` and `benchmark()` options. Because a variable was specified in `benchmark()`, the beta statistic, Jensen's alpha, and Treynor index can be calculated.

These assets show some variation in average returns, volatility, and summary indices. Beta is the coefficient on the benchmark asset in a regression. It measures comovement with the benchmark. A value near one indicates that the asset rises approximately in proportion to the benchmark. A value greater than one indicates that the asset moves more than one for one with the benchmark. A value near zero indicates that the asset does not have a tendency to comove with the benchmark. Negative values indicate assets that move against the benchmark: they tend to have good days when the benchmark has bad days, and vice versa, thus acting as a hedge.

For the assets above, ACME returns have fairly low beta risk. The three assets BAT, IRON, and DUNE have beta risk greater than one and thus are characterized as aggressive. They earn greater returns during "good" times when the benchmark is also delivering good returns, but they also earn even lower returns than the benchmark when the benchmark is earning poor returns. TYR has beta risk near one, indicating that its returns follow the market closely in magnitude.

Jensen's alpha is the intercept in a regression of the asset's return on the benchmark return. It has the interpretation of the return an asset provides, independently of the benchmark asset. For instance, in the above table, the ACME stock has an alpha of about 0.02. By contrast, the DUNE asset has a relatively high alpha of 0.22.

Treynor's index is the ratio of the excess return to beta risk and measures the "gain" in return an asset enjoys for each unit of beta risk it bears. For instance, most of the values in the above table are around 0.33, indicating that as beta risk increases by one unit, excess return rises by 0.33, or about one-third of a percent, per month.

The beta statistic and Jensen's alpha are identical to what would be obtained in a `finregress capm` financial regression using the same variables. Treynor's index uses the beta statistic to compute a return-to-market-risk ratio.

◁

▷ Example 3: Controlling the displayed statistics

The `statistics()` option allows finer control over which statistics are displayed. The mean and standard deviation are always reported. The option takes one or more of four keywords (`sharpe`, `jensen`, `treynor`, and `beta`) and displays only those statistics.

Using the same assets as the previous example, we can restrict the output to display only the Sharpe ratio and Jensen's alpha by typing

```
. finsummarize r_acme-r_tyr, rfrate(rf) benchmark(rmkt, adjust)
> statistics(sharpe jensen)
Financial summary statistics

Number of obs = 779
Sample: 1955m2 thru 2019m12
```

	Mean	Std. dev.	Sharpe ratio	Jensen's alpha
ACME	0.4517	1.1856	0.0443	0.0192
BAT	0.9052	5.4073	0.0927	0.1421
IRON	1.0214	6.5625	0.0941	0.1794
DUNE	1.0593	6.6088	0.0991	0.2147
TYR	0.7028	3.9642	0.0756	0.0433

```
Notes: Variables adjusted for risk-free rate rf.
       Variable rmkt used as benchmark asset and adjusted
       for risk-free rate rf.
```

◁

You may wish to compute standard errors and confidence intervals for the financial summary statistics. The `detail` option provides these statistics. The `post` option can then be used to post the results to `e()`, after which you can perform the usual tests using `test`, `testnl`, `lincom`, and `nlcom`.

▷ Example 4: Computing detailed statistics

We compute detailed summary statistics for two asset returns:

```
. finsummarize r_bat r_iron, rfrate(rf) benchmark(rmkt, adjust) detail
Financial summary statistics
Number of obs = 779
Sample: 1955m2 thru 2019m12
```

	Statistic	Robust std. err.	z	P>|z|	[95% conf.	interval]
r_bat						
mean	.9051807	.1936109	4.68	0.000	.5257104	1.284651
sd	5.407262	.1974776	27.38	0.000	5.020213	5.794311
sharpe	.0927026	.0369553	2.51	0.012	.0202714	.1651337
treynor	.3316203	.1276876	2.60	0.009	.0813572	.5818835
alpha	.1421061	.0420959	3.38	0.001	.0595997	.2246125
beta	1.521697	.0124204	122.52	0.000	1.497353	1.54604
r_iron						
mean	1.021351	.2349746	4.35	0.000	.5608094	1.481893
sd	6.562488	.2391546	27.44	0.000	6.093754	7.031223
sharpe	.0940612	.0368828	2.55	0.011	.0217724	.1663501
treynor	.3350978	.1268855	2.64	0.008	.0864068	.5837887
alpha	.1794485	.0464301	3.86	0.000	.0884472	.2704498
beta	1.852581	.0149286	124.10	0.000	1.823322	1.881841

```
Notes: Variables adjusted for risk-free rate rf.
       Variable rmkt used as benchmark asset and adjusted for risk-free rate
       rf.
```

The `detail` option provides estimates of the uncertainty around the financial summary statistics.

◁

Stored results

`finsummarize` stores the following in `r()`:

Scalars

`r(N)`	number of observations
`r(adjust)`	1 if benchmark asset return was adjusted, 0 otherwise

Macros

`r(stats)`	financial statistics list
`r(rfrate)`	risk-free rate
`r(benchmark)`	benchmark asset return variable
`r(wtype)`	weight type
`r(wexp)`	weight expression

Matrices

`r(table)`	table with all financial summary statistics

finsummarize with the detail option stores the following in r():

Scalars

r(adjust)	1 if benchmark asset return was adjusted, 0 otherwise
r(N_clust)	number of clusters
r(hac_lag)	HAC lag

Macros

r(rfrate)	risk-free rate
r(benchmark)	benchmark asset return variable
r(wtype)	weight type
r(wexp)	weight expression
r(vce)	*vcetype* specified in vce()
r(vcetype)	title used to label Std. err.
r(clustvar)	name of cluster variable
r(hac_kernel)	HAC kernel
r(nodfk)	nodfk, if specified

Matrices

r(b)	estimates
r(V)	variance–covariance matrix of the estimates

finsummarize with the post option stores the following in e():

Macros

e(properties)	b V

Matrices

e(b)	estimates vector
e(V)	variance–covariance matrix of the estimators

Methods and formulas

For each of the financial asset returns specified in finsummarize, the mean and standard deviation are computed using summarize. The Sharpe ratio, sharpe, for a given return r_t and a possibly time-varying risk-free rate r_t^f is computed by first forming the excess return series,

$$r_t^e = r_t - r_t^f$$

and then by computing the ratio of mean excess return μ_e to excess standard deviation σ_e,

$$\text{Sharpe} = \frac{\mu_e}{\sigma_e}$$

When the risk-free rate is zero, this expression reduces to the return's mean divided by its standard deviation. When the risk-free rate is constant, this expression becomes the excess return of the asset divided by the standard deviation of the asset.

The value reported as beta is the slope coefficient in a regression of the return and the asset specified in benchmark(). Jensen's alpha is the constant term in this regression. These computations are performed by finregress capm. When adjust is specified in the benchmark() option, the benchmark asset is also adjusted by the risk-free rate specified in rfrate().

The Treynor index, treynor, uses the slope coefficient β from finregress capm (see *Remarks and examples* in [FIN] **finregress capm** for more details), the mean of the returns μ_r, and the mean of the risk-free rate μ_r^f to get

$$\text{Treynor} = \frac{\mu_r - \mu_r^f}{\beta}$$

The `detail` option estimates all financial summary statistics jointly by the generalized method of moments and supports robust, cluster–robust, and HAC standard errors.

References

Andrews, D. W. K. 1991. Heteroskedasticity and autocorrelation consistent covariance matrix estimation. *Econometrica* 59: 817–858. https://doi.org/10.2307/2938229.

Gallant, A. R. 1987. *Nonlinear Statistical Models*. New York: Wiley. https://doi.org/10.1002/9780470316719.

Newey, W. K., and K. D. West. 1994. Automatic lag selection in covariance matrix estimation. *Review of Economic Studies* 61: 631–653. https://doi.org/10.2307/2297912.

Also see

[FIN] **finreturns** — Generate financial returns+

[FIN] **finregress capm** — Capital asset pricing model (CAPM)+

[TS] **tsset** — Declare data to be time-series data

finvalrisk — Value at risk[+]

[+]This command is part of StataNow.

Description

`finvalrisk` calculates value at risk (VaR) using the historical, normal-based, and model-implied methods.

Quick start

Compute historical and normal-based VaR for assets `p1` through `p4`

```
finvalrisk p1-p4
```

Same as above, but compute risk at the 1st and 5th percentiles

```
finvalrisk p1-p4, percentiles(1 5)
```

Use confidence level instead of percentile; calculate VaR at 90%, 95%, and 99% confidence levels

```
finvalrisk p1-p4, levels(90 95 99)
```

Compute ARCH-based VaR

```
finvalrisk p1-p4, model(arch , ar(1) arch(1) garch(1))
```

Menu

Statistics > Financial statistics > Value at risk

Syntax

```
finvalrisk varlist [ if ] [ in ] [ , options ]
```

options	Description
Main	
percentiles(*numlist*)	compute VaR for specified percentiles; default is percentiles(5)
levels(*numlist*)	compute VaR for specified confidence levels
altdef	use alternative percentiles formula for historical estimate
model(*model_spec*)	specify conditional mean model or conditional variance model or both
novarlabel	display variable names rather than variable labels

You must tsset your data before using finvalrisk; see [TS] **tsset**.

collect is allowed; see **[U] 11.1.10 Prefix commands**.

model_spec	Description
arima[, *arima_options*]	fit an autoregressive integrated moving-average (ARIMA) model
ewma [#][, noconstant]	fit an exponentially weighted moving-average (EWMA) model
arch[, *arch_options*]	fit an autoregressive conditional heteroskedasticity (ARCH) model

Options

Main

percentiles(*numlist*) specifies the percentiles at which VaR should be computed. Specifying percentiles(1) gives the first (1%) percentile. Specifying percentiles(0.1) gives the 0.1% percentile. The default is percentiles(5). percentiles() may not be specified with levels().

levels(*numlist*) specifies the confidence levels at which VaR should be computed. levels(90) specifies that the VaR should be computed using a 90% confidence level; this is equivalent to specifying percentiles(10). levels() may not be specified with percentiles().

altdef uses an alternative formula for calculating percentiles when the historical estimate is requested. The default method is to invert the empirical distribution function by using averages, $(x_i + x_{i+1})/2$, where the function is flat (the default is the same method used by summarize; see [R] **summarize**). The alternative formula uses an interpolation method. See *Methods and formulas* in [D] **pctile**.

model(*model_spec*) calculates VaR based on a model. Three models are supported: arima, ewma, and arch.

model(arima[, *arima_options*]) specifies that returns follow an ARIMA process. All options allowed with the arima command are allowed; see [TS] **arima**.

model(ewma [#][, noconstant]) specifies that the variance follows an EWMA model, $\sigma_t^2 = \omega + \lambda\sigma_{t-1}^2 + (1-\lambda)e_{t-1}^2$, where e_t is the error term. The parameter λ can be set to #. If it is not set, it is estimated. noconstant suppresses the parameter ω; otherwise, it is estimated.

model(arch[, *arch_options*]) specifies that returns follow an ARCH process. All options allowed with the arch command are allowed; see [TS] **arch**. Of particular interest are the three distributions of the error process available for arch: normal, Student's t, and the generalized error distribution.

novarlabel specifies that the variable names be displayed instead of the variable labels. By default, variable labels are used to label assets in the output.

Remarks and examples

Remarks are presented under the following headings:

Introduction

In financial analysis, it is often just as important to quantify potential losses as it is to summarize average behavior. Risk management methods seek to characterize low-probability, highly unfavorable events. The notion of VaR provides one measure of risk that is commonly used in practice.

VaR is a measure of potential loss on the value of an asset or a portfolio over a specified length of time for a given confidence level. It is often expressed as a number representing an amount of loss (or percentage loss) at a particular confidence level for a particular period. If the VaR on a portfolio is \$100 at a 95% confidence level at a one-month horizon, then there is a 5% chance of losing \$100 or more by the end of the month. The same portfolio might have a \$300 VaR at the 99% confidence level at one month, meaning we expect a 1% chance of losing \$300 or more by the end of the month. Put another way, losses more extreme than the specified VaR can still occur but should occur with less than the chosen percentile probability over the desired time frame.

The term "risk" can encompass multiple types of measurable uncertainty. Because VaR uses the observed historical performance of an asset or portfolio to assess risk, it is sometimes said to capture "market" risk and not, for example, counterpart, operational, or model risk (Boffelli and Urga 2016, chap. 5).

VaR can be thought of as an estimate of one or more quantiles of the distribution of asset returns. To estimate the quantiles, a model of potential returns on the asset or portfolio is required. Historical returns allow the model parameters to be fit to data. Many models have been proposed. The simplest is to use historical percentiles of the return distribution. Another approach is to estimate the parameters of a normal distribution based on historical returns and use the normal quantiles to compute VaR. This method, however, assumes time-invariant volatility, as well as normality of returns, neither of which may hold. Model-based approaches can be used to mitigate against the constant conditional mean assumption and the constant conditional variance assumption.

`finvalrisk` implements five models for measuring VaR:

1. Historical quantiles.
2. Normal quantiles based on the parameters estimated from historical data.
3. Quantiles derived from an ARIMA model for the mean of returns and a normally distributed error structure.
4. Quantiles derived from an EWMA model for the conditional variance.

5. Quantiles derived from an ARCH model for the conditional mean and conditional variance, with a choice of three error families for the error term: normal, Student's t, and the generalized error distribution.

We stress that all of these procedures rely on the assumption that past returns are a useful basis for forecasting the distribution of future returns.

Historical and normal-based VaR

The historical method of VaR uses observed quantiles of the distribution of returns. The normal-based approximation uses the mean and variance of the distribution of returns to compute a normal quantile.

We will compute the historical VaR and normal-based VaR for some fictional assets. The object of interest is the monthly percentage return on the asset, so the VaR will be a number like 1.5% at the 95% confidence level, meaning that there is a 5% chance of a loss of 1.5% or more. This VaR is sometimes referred to as a 5% VaR (based on the percentile) and sometimes as a 95% VaR (based on the confidence level). We follow Hurn et al. (2020) and call it a 5% VaR.

We can convert the VaR figures into dollar amounts by simply multiplying the percentage loss by the value of the asset.

Example 1: Historical return distribution

This example explores VaR graphically and numerically. We import some fictional stock prices. The data consist of the end-of-month price for 25 fictional publicly traded firms and range from 1955 through 2019. We use `finreturns` to generate monthly returns.

```
. use https://www.stata-press.com/data/r19/finex
(Fictional stock price data)
. quietly finreturns acme-tks, simple(r_) multiply(100)
```

`finreturns` created a set of variables `r_acme`, `r_bat`, etc, one for each asset. These variables contain the monthly simple return, multiplied by 100. So a value of 1.5 indicates that the asset grew in value by 1.5% in that month.

Somewhat arbitrarily, we choose asset `r_aaa` and observe its characteristics.

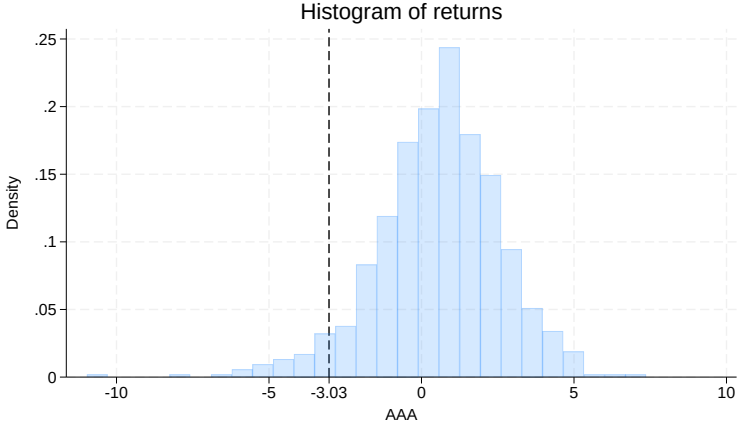
```
. summarize r_aaa, detail
                            AAA
-------------------------------------------------------------
      Percentiles      Smallest
 1%    -5.246088      -10.97514
 5%    -3.031731      -8.011154
10%    -1.930579      -6.801289       Obs                 779
25%    -.5408238      -6.047291       Sum of wgt.         779
50%     .7043946                      Mean           .5583062
                       Largest        Std. dev.       2.04691
75%     1.851859       5.240069
90%     2.941336       5.536678       Variance       4.189842
95%      3.61595       6.112684       Skewness      -.6100604
99%     4.996573       7.364605       Kurtosis       4.899915
```

The mean return for this asset is 0.56% per month, about one half of one percent per month, or about 6.9% per year, annualized using $1.0056^{12} - 1$. The percentiles give empirical quantiles of the distribution. We see that 1% of the time, this stock loses 5.25% or more of its value in a month. The percentiles reported

in the table are negative, representing losses, but the VaR is typically reported as a positive value. Thus, the asset's 1% VaR is 5.25%. Similarly, 5% of the time, the asset loses 3.03% or more in a month. Its 5% VaR is 3.03%.

The same information can be conveyed graphically, with a vertical line marking the 5% quantile of the distribution:

```
. histogram r_aaa, title("Histogram of returns") xline(-3.032)
> xlabel(-10 -5 -3.03 0 5 10) color(stblue%20)
(bin=27, start=-10.975141, width=.67924986)
```

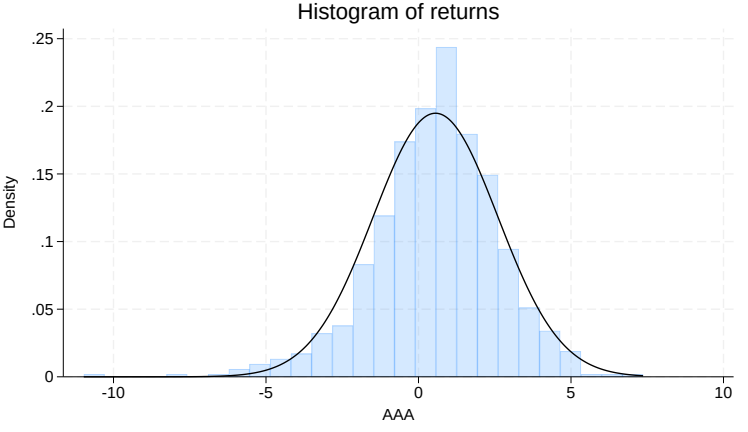


The observed returns are on average positive, though with a substantial fraction of months with negative return. The 5% VaR line marks the point at which 5% of returns are at that level of return or lower. We see there is a somewhat long left tail, reminding us that the 5% VaR is not the minimum return: 5% of returns are worse than it and indeed can be much worse.

Example 2: Normal-based VaR

Continuing with example 1, we see the normal-based approximation of VaR fits a normal distribution to the observed mean and variance of the returns:

```
. histogram r_aaa, title("Histogram of returns") normal color(stblue%20)
(bin=27, start=-10.975141, width=.67924986)
```



We could now compute the implied 5% normal quantile. Instead, we ask `finvalrisk` to do it for us. `finvalrisk` takes a variable list, which contains only `r_aaa` in this case. By default, the 5% VaR is reported, which corresponds to the 95% confidence level.

```
. finvalrisk r_aaa
Value-at-risk percentiles
Number of obs = 779
Sample: 1955m2 thru 2019m12
```

	Empirical	Normal
5%		
AAA	-3.031731	-2.808562

Both the historical and normal-based quantiles are reported. The normal-based VaR is 2.8%, a somewhat less extreme estimate than the empirical distribution suggests. Because our asset has a long left tail, the normal approximation underestimates VaR relative to the percentile method.

Example 3: Specifying multiple assets and thresholds

Continuing with the previous example, we now explore VaR for several assets and several thresholds. We extend the exploration to include a simple portfolio of assets. Using `finportfolio`, we can take an equally weighted average of four of the assets and call it `p1`.

```
. finportfolio equal r_aaa-r_cph, generate(p1, label("Portfolio"))
Equally weighted portfolio
Number of obs = 779
Sample: 1955m2 thru 2019m12
```

	Weight
AAA	.25
AFH	.25
ARD	.25
CPH	.25

```
Portfolio return    = 0.4809
Portfolio std. dev. = 1.6167
Risk-free rate      = 0.0000
Sharpe ratio        = 0.2975
```

Now we specify the `percentiles()` option to compute the 5% and 1% VaRs (or, equivalently, VaRs at the 95% and 99% confidence levels) for the individual assets and the portfolio.

```
. finvalrisk r_aaa-r_cph p1, percentiles(5 1)
Value-at-risk percentiles
Number of obs = 779
Sample: 1955m2 thru 2019m12
```

	Empirical	Normal
5%		
AAA	-3.031731	-2.808562
AFH	-3.629369	-3.287768
ARD	-2.758431	-2.796273
CPH	-2.991949	-3.005515
Portfolio	-2.366248	-2.178334
1%		
AAA	-5.246088	-4.203519
AFH	-5.519961	-4.923469
ARD	-4.401581	-4.071673
CPH	-5.37588	-4.426132
Portfolio	-4.672417	-3.280123

The output table has two blocks, one for each VaR threshold. At the 5% level, the four individual assets have VaRs ranging from 3.63% to 2.76%. The equally weighted portfolio achieves a less extreme VaR, 2.37%. However, portfolios are not guaranteed to beat individual assets, especially at the tails. At the 1% VaR threshold, the portfolio has a VaR of 4.67%, but one of the assets (ARD) has a slightly lower VaR of 4.40%.

Model-based VaR

So far, we have used the historical distribution of returns and a normal-based approximation to that distribution to compute VaR. A more sophisticated approach involves estimating the conditional mean or conditional variance of returns, or both, as time-varying objects. Doing so generates a VaR that can fluctuate over time as the behavior of returns changes. Three models are available. The ARIMA model is a model for the conditional mean. Intuitively, if stock returns have any predictability on average, then the expected mean return can shift over time. If the expected mean is unusually high or low in a given period, it will shift the whole distribution and thus shift VaR. The EWMA model and ARCH model are models for the conditional variance. Intuitively, many stock returns exhibit clusters of high or low volatility. In high-volatility periods, the distribution of returns is itself wider, thus increasing VaR; in low-volatility periods, the distribution of returns is narrower, reducing VaR.

Example 4: ARIMA for the mean

In this example, we continue with example 3 to explore VaR for the four sample assets and their equally weighted portfolio. We begin by asking whether a model for the average return adds any nuance to the VaR calculations. We consider a simple model for the mean where returns r_{it}'s are modeled based on their first and second lags and where errors e_{it}'s are assumed to be normally distributed.

$$r_{it} = \beta_0 + \beta_1 r_{i,t-1} + \beta_2 r_{i,t-2} + e_{it}$$
$$e_{it} \sim N(0, \sigma^2)$$

This is a simple AR(2) model, and its parameters β_0, β_1, β_2, and σ^2 can be estimated with `arima`. We fit this model to one of the assets, `r_aaa`.

```
. arima r_aaa, ar(1/2) nolog
ARIMA regression
Sample: 1955m2 thru 2019m12                     Number of obs     =        779
                                                Wald chi2(2)      =      34.22
Log likelihood = -1647.842                      Prob > chi2       =     0.0000
```

r_aaa	Coefficient	OPG std. err.	z	P>|z|	[95% conf.	interval]
r_aaa						
_cons	.5579395	.0869368	6.42	0.000	.3875466	.7283325
ARMA						
ar						
L1.	.1951391	.0337801	5.78	0.000	.1289313	.2613469
L2.	-.0676834	.0351207	-1.93	0.054	-.1365187	.001152
/sigma	2.006473	.0415384	48.30	0.000	1.925059	2.087887

```
Note: The test of the variance against zero is one sided, and the two-sided
      confidence interval is truncated at zero.
```

There is some mild forecastability: the first autoregressive lag coefficient is about 0.2, so when returns are 1% higher than their long-run average in a given period, then returns are expected to be 0.2% above their long-run mean in the following period.

To use finvalrisk to fit this AR(2) model to each asset and the portfolio, we specify the model() option, select a model, and provide any model-specific options. We obtain the following output:

```
. finvalrisk r_aaa-r_cph p1, model(arima, ar(1/2))
Fitting ARIMA models ...
Value-at-risk percentiles
Model: ARIMA
Error family: Normal
Number of obs = 779
Sample: 1955m2 thru 2019m12
```

	Empirical	Normal	ARIMA historical	ARIMA one step
5%				
AAA	-3.031731	-2.808562	-2.742248	-2.922465
AFH	-3.629369	-3.287768	-3.263831	-3.238124
ARD	-2.758431	-2.796273	-2.791969	-2.763895
CPH	-2.991949	-3.005515	-2.964135	-2.702158
Portfolio	-2.366248	-2.178334	-2.106934	-2.028872

```
Note: One-step-ahead value at risk for 2020m1.
```

The output now has four columns, one for each VaR method. As before, the first column reports the historical quantile, and the second column reports the normal-based quantile.

The third column reports the quantile for the model-based VaR averaged across the sample. The final column provides the quantile for the one-step-ahead VaR forecast for one period out from the sample period. For example, these fictional data end in December 2019, so the one-step-ahead quantile is reported for January 2020.

Example 5: EWMA for the variance

The EWMA model allows a time-varying error variance. Specifically, instead of assuming that the variance is constant at σ^2, the error variance is allowed to depend on past error variance and the most recent error value:

$$\sigma_t^2 = \omega + \lambda\sigma_{t-1}^2 + (1-\lambda)e_{t-1}^2$$

The time-varying variance allows for periods of high volatility and low volatility. In a period of high volatility, the distribution of returns is more spread out, and VaR will be more extreme than average. In periods of calm, the distribution of returns is less spread out, and VaR will be less extreme than average.

`finvalrisk` implements this method in the `model(ewma [#])` option, where the optional number is the weight (or shape parameter) λ. If a weight is not given, it is calculated optimally from the data.

```
. finvalrisk r_aaa-r_cph p1, model(ewma 0.95)
Fitting EWMA models ...
Value-at-risk percentiles
Model: EWMA
Error family: Normal
Shape parameter = 0.95
Number of obs = 779
Sample: 1955m2 thru 2019m12
```

	Empirical	Normal	EWMA historical	EWMA one step
5%				
AAA	-3.031731	-2.808562	-2.92945	-2.431956
AFH	-3.629369	-3.287768	-3.35059	-2.619925
ARD	-2.758431	-2.796273	-2.905166	-2.423582
CPH	-2.991949	-3.005515	-3.142034	-3.021991
Portfolio	-2.366248	-2.178334	-2.277888	-1.991785

```
Note: One-step-ahead value at risk for 2020m1.
```

The header reports the EWMA shape parameter of 0.95, which we specified. In the output table, the third column gives the quantile corresponding to the historical average VaR from the EWMA model for each asset. The fourth column gives the one-step-ahead value for the period immediately following the last sample period.

The portfolio has an empirical historical VaR of 2.37%. The average EWMA estimated VaR is a little lower, 2.28%. But the one-step-ahead forecasted VaR for January 2020 is 2.0%, indicating that, given the most recent observations of 2019, January 2020 is predicted to be a quieter month.

Example 6: ARCH for the mean and variance

In this example, we fit an ARCH model of the conditional mean and conditional variance. We first do so by hand and compute time-varying VaR by hand to provide intuition. We fit a generalized ARCH$(1,1)$ model with the arch command, using the arch(1) and garch(1) options.

```
. arch r_aaa, ar(1) arch(1) garch(1) nolog
ARCH family regression -- AR disturbances
Sample: 1955m2 thru 2019m12                     Number of obs     =        779
                                                Wald chi2(1)      =      13.22
Log likelihood = -1636.907                      Prob > chi2       =     0.0003
```

r_aaa	Coefficient	OPG std. err.	z	P>|z|	[95% conf.	interval]
r_aaa						
_cons	.5896113	.0860708	6.85	0.000	.4209157	.758307
ARMA						
ar L1.	.1525666	.041967	3.64	0.000	.0703129	.2348204
ARCH						
arch L1.	.0883585	.0241089	3.66	0.000	.0411059	.1356112
garch L1.	.8003264	.0692668	11.55	0.000	.664566	.9360868
_cons	.4555743	.2151189	2.12	0.034	.0339489	.8771997

The ARCH and GARCH terms are both positive, indicating that the variance of the series exhibits some positive autocorrelation, so periods of high and low variance are clustered together.

To compute VaR by hand, we use predict after arch to predict the conditional mean and variance. Then we build VaR using the normal error distribution assumption.

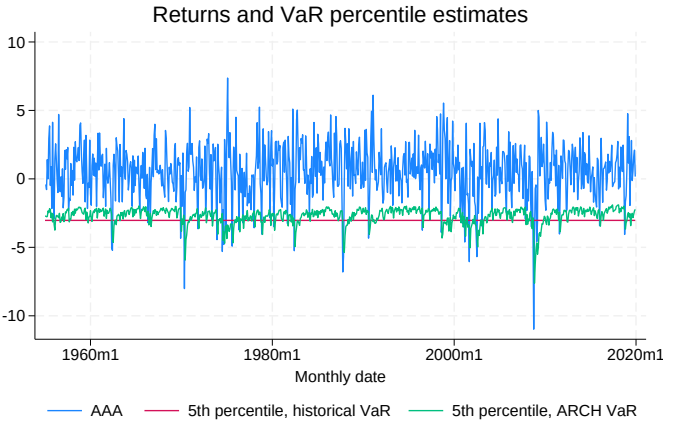
```
. predict raaa_mean, xb
. predict raaa_variance, variance
. generate raaa_VaR_arch = raaa_mean + invnormal(0.05) * sqrt(raaa_variance)
```

These three steps compute percentiles corresponding to the 5% ARCH-based VaR. We can now graph it, along with the percentile corresponding to historical 5% VaR as a benchmark. We first generate a variable with the 5th percentile for the historical VaR, and then we add some variable labels:

```
. label variable raaa_VaR_arch "5th percentile, ARCH VaR"
. generate raaa_VaR_hist = -3.03
. label variable raaa_VaR_hist "5th percentile, historical VaR"
```

The following graph plots returns over time, along with loss corresponding with ARCH and historical VaR.

```
. tsline r_aaa raaa_VaR_hist raaa_VaR_arch, legend(rows(1))
> title("Returns and VaR percentile estimates")
```



We see that the volatility of the series itself varies over time, with some periods of relatively high volatility and other periods of relative calm. The potential loss based on the historical VaR is a straight line at the 5% quantile of the distribution. The potential loss based on the ARCH-based VaR adapts over time to the recent conditions of the series. For example, when returns are especially volatile, potential loss becomes more negative, reflecting higher risk in those time periods. We next use `finvalrisk` to perform these calculations.

```
. finvalrisk r_aaa-r_cph p1, model(arch, ar(1) arch(1) garch(1))
Fitting ARCH models ...
Value-at-risk percentiles
Model: ARCH
Error family: Normal
Number of obs = 779
Sample: 1955m2 thru 2019m12
```

	Empirical	Normal	ARCH historical	ARCH one step
5%				
AAA	-3.031731	-2.808562	-2.701858	-2.433388
AFH	-3.629369	-3.287768	-3.166423	-2.750784
ARD	-2.758431	-2.796273	-2.773078	-2.344072
CPH	-2.991949	-3.005515	-2.923198	-2.515059
Portfolio	-2.366248	-2.178334	-2.048298	-1.665521

```
Note: One-step-ahead value at risk for 2020m1.
```

The historical and normal-based VaR are the same as we have already encountered. The 5% quantile for the ARCH-based VaR for the historical sample and for the one-step-ahead prediction are reported in the third and fourth columns, respectively. Notice that while the historical ARCH average VaR for the portfolio is 2.05%, the one-step-ahead conditional VaR is much smaller at 1.67%, indicating that the upcoming period is predicted to be less volatile than the historical average. This prediction would be driven by the underlying ARCH model.

Alternative error distributions

Although we have so far allowed time-varying variance, the underlying model for the error terms remains a normal distribution. The `arch` command allows three distributions: normal, Student's t, and the generalized error distribution. The last two both have shape parameters, which can be controlled manually or estimated. The shape parameter of the t distribution is its degrees of freedom. The shape parameter for the generalized error distribution controls the thickness of the tails. When the shape parameter of the generalized error distribution is less than two, its tails are fatter than a normal distribution; when the shape parameter is greater than two, the tails are thinner than the normal distribution.

Example 7: ARCH with GED errors

To specify the t distribution, we would use `distribution(t)`. But we are interested in the generalized error distribution, so we type `distribution(ged)`. We also specify `percentiles(1)` because we are interested in the 1% VaR.

```
. finvalrisk r_aaa-r_cph p1,
> model(arch, ar(1) arch(1) garch(1) distribution(ged)) percentiles(1)

Fitting ARCH models ...

Value-at-risk percentiles

Model: ARCH
Error family: Generalized error
Shape parameter = 1.60

Number of obs = 779
Sample: 1955m2 thru 2019m12
```

	Empirical	Normal	ARCH historical	ARCH one step
1%				
AAA	-5.246088	-4.203519	-4.344637	-3.953778
AFH	-5.519961	-4.923469	-5.097832	-4.542212
ARD	-4.401581	-4.071673	-4.054237	-3.442605
CPH	-5.37588	-4.426132	-4.57175	-4.116522
Portfolio	-4.672417	-3.280123	-3.434237	-3.003905

```
Note: One-step-ahead value at risk for 2020m1.
```

In the output header, Stata reports we are using the generalized error distribution. The shape parameter is estimated separately for each return or portfolio, and the reported shape parameter is the average of the estimated shape parameters. The average here is 1.60, which indicates that the returns and portfolios considered here are estimated to have fatter tails than the normal distribution would predict. VaR calculations, both historical average and one-period-ahead, account for these fatter tails. For example, the historical average ARCH VaR for asset AAA is 4.34%, whereas the normal distribution approach has VaR of 4.20%.

You may wish to set the shape parameter instead of estimating it. To do so, you can ask for an ARCH model with a specific `distribution(ged #)`.

Example 8: Fixing the shape parameter

We specify a relatively small value of the shape parameter, 1.20, to see how it affects VaR estimation. Unlike the previous example, the shape parameter is fixed and fixed to the same value for all assets.

```
. finvalrisk r_aaa-r_cph p1,
> model(arch, ar(1) arch(1) garch(1) distribution(ged 1.2)) percentiles(1)

Fitting ARCH models ...

Value-at-risk percentiles

Model: ARCH
Error family: Generalized error
Shape parameter = 1.20

Number of obs = 779
Sample: 1955m2 thru 2019m12
```

	Empirical	Normal	ARCH historical	ARCH one step
1%				
AAA	-5.246088	-4.203519	-4.794479	-4.403476
AFH	-5.519961	-4.923469	-5.76262	-5.182215
ARD	-4.401581	-4.071673	-4.901967	-4.184606
CPH	-5.37588	-4.426132	-5.152057	-4.695892
Portfolio	-4.672417	-3.280123	-3.63922	-3.222156

```
Note: One-step-ahead value at risk for 2020m1.
```

As expected, specifying an error distribution with fatter tails leads to an increased estimated downside risk, and VaR for the ARCH models grows. For instance, a 1% VaR is now associated with a 4.80% decline in the AAA asset returns on average, compared with the normal distribution estimate of 4.20%.

Stored results

`finvalrisk` stores the following in `r()`:

Scalars

`r(N)`	number of observations
`r(tmin)`	minimum time
`r(tmax)`	maximum time
`r(shape)`	shape parameter for models `arch` and `ewma`
`r(altdef)`	1 if `altdef` is specified, 0 otherwise

Macros

`r(model)`	`arch` or `arima` or `ewma`
`r(modelopts)`	model-specific options
`r(tmins)`	formatted minimum time
`r(tmaxs)`	formatted maximum time
`r(percentiles)`	requested percentiles
`r(levels)`	requested confidence levels

Matrices

`r(table)`	matrix containing empirical, normal, and model-based value-at-risk statistics

Methods and formulas

Methods and formulas are presented under the following headings:

Historical quantiles

`finvalrisk` uses the `_pctile` command to estimate historical quantiles. The default formula for percentiles is as follows. For each series x_t, let $x_{(j)}$ refer to the x in ascending order for $j = 1, 2, \ldots, T$. Hence, $x_{(j)}$ is the jth largest x. To obtain the pth percentile, which we will denote as $x_{[p]}$, let $P = Tp/100$. Let i be the integer floor of $Tp/100$. The pth percentile is then

$$x_{[p]} = \begin{cases} \dfrac{x_{(i)} + x_{(i+1)}}{2} & \text{if } i = P \\ x_{(i+1)} & \text{otherwise} \end{cases}$$

When the `altdef` option is specified, the following alternative definition is used. Let i be the integer floor of $(T+1)p/100$; that is, i is the largest integer $i \leq (T+1)p/100$. Let h be the remainder $h = (T+1)p/100 - i$. The pth percentile is then

$$x_{[p]} = (1-h)x_{(i)} + hx_{(i+1)}$$

where $x_{(0)}$ is taken to be $x_{(1)}$ and $x_{(T+1)}$ is taken to be $x_{(T)}$. See [D] **pctile** for further information on `_pctile`.

The underlying modeling assumption is that the historical distribution of returns is stable and that the next period's return is a random draw from the historical distribution. Thus, the p% VaR is the pth quantile of the historical distribution.

Normal quantiles

Estimates of the normal quantile are a function of the estimated mean, the estimated standard deviation, and a normal density. The formula for the pth percentile for variable x_t is

$$x_p = \hat{\mu} + \hat{\sigma}\Phi^{-1}(p/100)$$

where $\hat{\mu}$ is the estimated mean of x and $\hat{\sigma}$ is the estimated standard deviation of x. The function $\Phi^{-1}(p)$ is the inverse cumulative standard normal distribution; if $\Phi(z) = p$, then $\Phi^{-1}(p) = z$.

The underlying modeling assumption is that returns are independent, identically distributed draws from a normal distribution. Once the mean μ and standard deviation σ have been estimated, the p% VaR is the pth quantile of the normal distribution. This method may understate VaR if the actual distribution is nonnormal or if the mean or variance of the distribution changes over time.

Model-based quantiles

Model-based quantiles use a model to obtain potentially time-varying estimates of the mean return $\hat{\mu}_t$ and the standard deviation $\hat{\sigma}_t$, then combine these estimates with a distribution for the model errors. Because the conditional mean and conditional standard deviation can be time varying, the quantiles of the expected return distribution, and thus VaR, can also be time varying.

Regardless of the model used, `finvalrisk` reports two numbers: the average and one-step-ahead VaR. The average model-based VaR is the average of the period-by-period VaR estimates. The one-step-ahead VaR is the VaR predicted for one period after the sample ends, using the one-step out-of-sample estimate of the conditional mean and conditional standard deviation.

ARIMA

`finvalrisk` with the `model(arima , ...)` option fits an ARIMA model for each series x_t and uses the model to obtain quantile estimates.

The conditional mean $\hat{\mu}_t$ and conditional standard deviation $\hat{\sigma}_t$ are predicted each period using the fitted model. Then a value-at-risk estimate for each period is constructed from the estimated mean, estimated standard deviation, and the normal distribution.

ARCH

`finvalrisk` with the `model(arch, ...)` option estimates an ARCH model for each series x_t and uses the model to form quantile estimates.

One of the simpler ARCH models is the generalized ARCH(1,1), in which the conditional variance evolves over time via

$$\sigma_t^2 = \omega + \alpha e_{t-1}^2 + \beta \sigma_{t-1}^2$$

where (ω, α, β) are parameters to be estimated and e_t^2 is the squared error from the model. Many more complicated models can be fit; see [TS] **arch**.

The conditional mean $\hat{\mu}_t$ and conditional standard deviation $\hat{\sigma}_t$ are predicted each period using the fitted model. Then a value-at-risk estimate for each period is constructed from the estimated mean, estimated standard deviation, and one of three distributions. The average model-based quantile is the average of the time series of VaR estimates. The available distributions are a normal distribution, a t distribution, and a generalized error distribution. The t distribution and generalized error distribution each depend on a shape parameter. This parameter can be fixed or estimated.

EWMA

The EWMA process for the variance is

$$\sigma_t^2 = \omega + \lambda \sigma_{t-1}^2 + (1-\lambda) e_{t-1}^2$$

The parameter λ can be fixed or estimated. The parameter ω can be estimated or fixed to 0 with the `noconstant` option. Then quantiles are computed from a normal distribution with mean $\hat{\mu}$ and standard deviation $\hat{\sigma}_t$.

References

Boffelli, S., and G. Urga. 2016. *Financial Econometrics Using Stata*. College Station, TX: Stata Press.

Hurn, S., V. L. Martin, P. C. B. Phillips, and J. Yu. 2020. *Financial Econometric Modeling*. Oxford: Oxford University Press.

Also see

[FIN] **finportfolio** — Financial portfolio selection+

[FIN] **finreturns** — Generate financial returns+

Glossary

annualization. Annualization is the process of calculating the annual equivalent of returns that were measured at higher frequencies, such as daily or monthly returns. To annualize the monthly return for a particular month, we compound 12 times. To annualize a daily return, we must consider how many days a year the asset in question actually generates returns, that is, how many trading days there are in a year.

benchmark asset. A reference investment used to evaluate the performance of assets or portfolios is called the benchmark asset. These are typically market indices, like the S&P 500, or "risk-free" investments, like US treasuries.

beta. An asset's "market beta" is a coefficient measuring how sensitive the returns of the asset are to movements in the overall market. This is the slope of excess market returns in a capital asset pricing model.

beta risk. Beta risk is the risk associated with the covariance between an asset and an independent variable. In a capital asset pricing model, beta risk is captured by the regression coefficient on that independent variable.

β risk. See *beta risk*.

capital asset pricing model. A capital asset pricing model (CAPM) is a model for the relationship between an asset's return and independent variables. According to the CAPM, the return we can expect from an asset is equal to the risk-free rate, plus a risk premium that is proportional to the mean of the independent variables in the model.

CAPM. See *capital asset pricing model*.

dollar return. The change in the price of an asset between two periods is sometimes called the dollar return, that is, $P_t - P_{t-1}$.

excess return. It is often desirable to study the returns an asset generates in excess of the returns generated by some benchmark asset. These are called excess returns.

factor. In the context of asset-pricing models, factors are sources of risk considered in the model. Factors such as market volatility and firm size serve as independent variables in such models.

federal funds rate. The interest rate at which banks in the United States lend reserves to each other overnight. The rate is set by the Federal Reserve, and it is a key benchmark for short-term interest rates in all debt markets.

financial data. Data containing characteristics of stocks, bonds, currency, portfolios, and other series of financial interest. Prices, returns, interest rates, dividends, and exchange rates are all examples of financial data.

financial statistics. The branch of statistics that provides methods for the analysis of financial data. The methods of financial statistics are used to explain and predict the performance of various assets and portfolios. Financial summary statistics provide information on returns and risk. In the analysis of stocks, for instance, one may estimate the market beta, value at risk, and various risk-return ratios. The CAPM and Fama–MacBeth regressions provide model-based financial statistics. Additional time-series models such as autoregressive integrated moving-average models and autoregressive conditional heteroskedasticity models provide statistics that are useful for analysis and forecasting of financial data.

gross return. The ratio of the price of an asset between two periods is called the gross return, that is, P_t/P_{t-1}.

hedge. A hedge is an investment whose purpose is to counteract the risk of another investment. For example, one may counterbalance the risk of investing in the fossil fuel industry by also investing in clean-energy stocks. To hedge an investment means to counterbalance its risk in this manner. An asset is called a hedge against a benchmark in a CAPM regression if its slope coefficient is less than zero; see *Remarks and examples* in [FIN] **finregress capm**.

inflation. Inflation is the general increase in prices that occurs over time in an economy, reducing the purchasing power of currency.

Jensen's alpha. The intercept in a capital asset pricing model measures the average return in excess of market returns and is called Jensen's alpha.

log return. The change in the log of the price of an asset between two periods is called the log return, that is, $\log P_t - \log P_{t-1}$.

market beta. See *beta*.

multifactor model. A capital asset pricing model where factors other than market risk are additionally included is called a multifactor model.

portfolio. A collection of assets held by an investor is called a portfolio. The returns and risk of a portfolio depend on the returns and risk of its component assets, as well as on the correlation between its component assets.

portfolio return. The portfolio return is a weighted average of the returns of individual assets in a portfolio.

portfolio risk. Portfolio risk is a measure of the variability of portfolio return.

portfolio weights. Portfolio weights indicate the fraction of total investment allocated to each asset in a portfolio.

price of risk. Different risky assets compensate investors differently for the risk they entail; that is, they have different risk premiums. The price of risk is a summary measure of how well or poorly risk is compensated in a particular market. Specifically, price of risk is the factor by which expected returns grow with the so-called beta risk. See *Fama–Macbeth regression* in [FIN] **fin** for details.

return. Return is the value an investment gains or loses over a period of time.

risk. The risk of an investment is the variability of its returns, that is, any measure that reflects the probability that returns will deviate from expected returns.

risk-free rate. A rate of return that can be obtained with negligible or no risk, such as the returns of sovereign debt in stable economies, is called a risk-free rate.

risk premium. A rational investor will take on risk only if they consider this risk to be "worth it", that is, compensated in terms of the returns they expect to obtain. The risk premium is the difference between the expected returns of a risky investment and the expected returns of investing in a risk-free asset. Investors will consider allocating wealth to risky assets only when the risk premium is positive, that is, when the risky assets offer expected returns above what one may obtain in the market without incurring risk. In a Fama–MacBeth regression, the intercept term can be interpreted as the risk premium on an asset with zero beta risk.

Sharpe ratio. A ratio of excess returns to the variability of an asset, or a ratio of excess returns to the variability of excess returns of an asset, is a popular performance measure. When variability is represented by the standard deviation of returns or of excess returns, the performance measure is called a Sharpe ratio.

short selling. When the price of an asset is expected to fall, investors may borrow and then sell the asset to buy it back at a later date and a lower price. They can then deliver the asset to the original lender and pocket the price difference. This strategy is called short selling.

simple return. The proportional change in the price of an asset between two periods is called the simple return, that is, $(P_t/P_{t-1}) - 1$.

Treynor index. A ratio of excess returns to the variability of an asset, or a ratio of excess returns to the variability of excess returns of an asset, is a popular performance measure. When variability is represented by the beta of a capital asset pricing model, the performance measure is called a Treynor index.

value at risk. Value at risk (VaR) is a measure of potential loss on the value of an asset or portfolio over a specified length of time for a given confidence level. It is often expressed as a number representing an amount of loss (or percentage loss) at a particular confidence level for a particular period. If the VaR on a portfolio is \$100 at a 95% confidence level at a one-month horizon, then there is a 5% chance of losing \$100 or more by the end of the month.

volatility. Volatility is the degree to which asset prices fluctuate over time, as measured, for example, by their variance or standard deviation.

Subject and author index

See the combined subject index and the combined author index in the *Stata Index*.