Performing trade costs analysis with the `tradeCosts` package

Aaron Schwartz and Luyi Zhao

June 25, 2007

Abstract

For those who frequently engage in trading securities in financial markets, trade costs cannot be overlooked. Without an understanding of the costs incurred during trading returns can never reach their full potential. The `tradeCosts` package provides an easy-use to use set of tools for analyzing trade costs by generating automated summaries and PDF reports of trade costs from raw trading data. The summaries and reports generated allow a user to quickly understand how far their trades executed were from a user-specified benchmark price.

1 Introduction

For institutions and individuals that frequently trade securities in financial markets trade costs are an important expense to be considered. A money manager ignoring trade costs can hinder the performance of a portfolio by not understanding the costs of trades being made. We define trading costs in our package based on the work of Kissell and Glantz's *Optimal Trading Strategies* [1]. As described in their work, trade costs can be thought of to be composed of several distinct elements: commission, fees, spreads, price appreciation, market impact, timing risk, and opportunity cost. Currently, however, the package focuses solely on slippage, which we refer to in this release of the package as cost, as an ex-post measurement of the quality of executions relative to a benchmark price.

We rank trades by percent cost, which we define in more detail as:

$$\%\ \text{cost} = 100 * \frac{\text{cost}}{\text{market value of executed shares}},$$

where

$$\text{market value of executed shares} = \text{execution quantity} * \text{execution price},$$

and

$$\text{cost} = \text{side adjustment} * \text{execution quantity} * (\text{execution price} - \text{benchmark price}).$$

Since we wish for the market value of executed shares to be unsigned, we assume that execution quantity is always positive regardless of the trade's side. The side adjustment variable is either or 1 or -1 and serves to adjust the sign of cost depending on the side of the trade. In the `tradeCosts` package, positive cost is considered bad while negative cost is good (Example: a negative percent cost on a buy order indicates that the security was purchased for less than the benchmark price). Thus for buy and cover orders side adjustment is 1, and for sell and short orders side adjustment is just -1. Also, total percent costs over a certain time period or security are calculated as the market value weighted average of individual trades.

Having the ability to quickly and automatically generate reports that provide information about how far trades were being executed from a benchmark price for a certain group of trades can yield valuable information for those trying to improve their returns. These reports and summaries can help users gain a big picture view of the trade costs for a group of trades and discover specific trades, time periods, or specific securities with unusually high or low trade costs. The package operates through a user interface function, `analyze.trade.costs` which displays summaries of the input trade data and/or generates PDF summaries.

2 Calculating Cost: Examples

We now go through simple examples of calculating slippage, or as we refer to it in this release, cost, for individual trades, a period of time, and a security.

2.1 Individual Trades

First, we examine the process of calculating cost and percent cost for individual trades. Suppose for March, 14 2007 security FOO had a volume weighted average price (VWAP) of \$3.13 and that on that day two shares of FOO were brought at the price of \$3.15. The price that we actually bought the securities at, \$3.15, is referred to as the execution price, and the VWAP will be the benchmark price for this case. The number of shares we bought, two, is referred to as the execution quantity. Also, as noted in the previous section, since this is a buy order the side adjustment factor for this order should be 1. According to our formula introduced in the introduction, we then calculate cost as:

$$\begin{aligned}
&\text{cost} = \text{side adjustment} * \text{execution quantity} * (\text{execution price} - \text{benchmark price}) \\
&\text{cost} = 1 * 2 * (3.15 - 3.13) \\
&\text{cost} = \$0.04.
\end{aligned}$$

Percent cost can then be easily calculated as:

$$\begin{aligned}
\%\ \text{cost} &= 100 * \frac{\text{cost}}{\text{market value of executed shares}} \\
\%\ \text{cost} &= 100 * \frac{\text{cost}}{\text{execution quantity} * (\text{execution price})} \\
\%\ \text{cost} &= 100 * \frac{0.04}{2 * 3.15} \\
\%\ \text{cost} &= 0.63\%
\end{aligned}$$

Notice that both percent cost and cost are positive which indicate that relative to the benchmark price, the trade described yielded a loss for whoever executed the trade. If FOO had been sold at \$3.13, then our calculations above would have yielded a cost of $-\$0.04$ and a percent cost of -0.63%. The negative percent cost and cost would have indicated that the transaction, relative to the VWAP benchmark price, was a good trade for whoever bought the two shares of FOO.

2.2 Security

In addition to calculating the cost of an individual trade we can also examine the total trading cost of a security. For example, suppose we have three trades of FOO stock, the details of which are summarized in the table below:

Period	Security Name	Side	Execution Price	Execution Quantity	VWAP
2/19/2007	FOO	buy	$2.23	5	$2.18
3/14/2007	FOO	buy	$3.15	2	$3.13
4/4/2007	FOO	sell	$3.51	5	$3.62
4/16/2007	FOO	short	$4.15	4	$4.28

In order to calculate the total cost of the security FOO we simply sum up the costs of each individual trade:

$$\begin{aligned}
&\text{cost} = \sum(\text{side adjustment} * \text{execution quantity} * (\text{execution price} - \text{benchmark price})) \\
&\text{cost} = 1 * 5 * (2.23 - 2.18) + 1 * 2 * (3.15 - 3.13) + -1 * 5 * (3.51 - 3.62) + -1 * 4 * (4.15 - 4.28) \\
&\text{cost} = 1.36
\end{aligned}$$

To obtain the total percent cost we divide the total cost by the total market value of all the executions as shown below:

$$\begin{aligned}
\%\ \text{cost} &= 100 * \frac{\text{total}}{\sum \text{market value of executed shares}} \\
\%\ \text{cost} &= 100 * \frac{1.36}{5 * 2.23 + 2 * 3.15 + 5 * 3.51 + 4 * 4.15} \\
\%\ \text{cost} &= 100 * \frac{1.36}{5 * 2.23 + 2 * 3.15 + 5 * 3.51 + 4 * 4.15} \\
\%\ \text{cost} &= 2.64\%
\end{aligned}$$

2.3 Period

Similarly, the total cost and percent cost can be found for a certain period of time. Consider the table of trades below:

Period	Security Name	Side	Execution Price	Execution Quantity	VWAP
3/14/2007	FOO	buy	$3.15	2	$3.13
3/14/2007	BAR	cover	$21.71	5	$22.00
3/14/2007	MOR	sell	$14.51	4	$14.28

Now instead of summing over the trades for a security we group the trades by period, in this case 3/14/2007, and find the total cost of the trades for 3/14/2007:

$$\begin{aligned}
&\text{cost} = \sum(\text{side adjustment} * \text{execution quantity} * (\text{execution price} - \text{benchmark price})) \\
&\text{cost} = 1 * 2 * (3.15 - 3.13) + 1 * 5 * (21.71 - 22.00) + -1 * 4 * (14.51 - 14.28) \\
&\text{cost} = -2.33
\end{aligned}$$

Like before, dividing by the sum of the market values of the executed trades yields the percent cost:

$$\% \text{ cost} = 100 * \frac{\text{total}}{\sum \text{market value of executed shares}}$$
$$\% \text{ cost} = 100 * \frac{1.36}{5 * 2.23 + 2 * 3.15 + 5 * 3.51 + 4 * 4.15}$$
$$\% \text{ cost} = 2.64\%$$

3 Trade Costs Analysis: An Example

Here we step through a trade costs analysis using the top-level function `analyze.trade.costs`. After understanding the required data, we explain the options available in `analyze.trade.costs`, call the function, and examine the summary output.

First, we will introduce the raw data sets included in this package: `trade`, a `data.frame` of trading data; `dynamic`, a `data.frame` of dynamic securities data; and `static`, a `data.frame` of static securities data. Function `analyze.trade.costs` takes in the trade, dynamic descriptive, and static descriptive data in separate `data.frames`. `trade` includes the IDs of the securities, the time period, side, execution quantity, the execution price:

```
> data(trade.mar.2007)
> head(trade.mar.2007, n = 1)

      period       id side exec.qty exec.price
5 2007-03-01 03818830    X    60600        1.6
```

`dynamic` is a `data.frame` that represents the dynamic descriptive data and includes the IDs of the securities, the time period, and a benchmark price:

```
> data(dynamic.mar.2007)
> head(trade.mar.2007, n = 1)

      period       id side exec.qty exec.price
5 2007-03-01 03818830    X    60600        1.6
```

Finally, we have the `static`, the static descriptive data, which has the IDs and symbols (an alternative identification from ID) of the securities:

```
> data(static.mar.2007)
> head(static.mar.2007, n = 1)

           id symbol      name
XSNX 98385L10   XSNX Xsunx Inc
```

Once the data is loaded with the correct columns, it is time to call `analyze.trade.costs` This user-level function returns either a text summary generated in the `R` environment or a PDF report. A `tradeCostsResults` object is also returned and is explained below in section 4. Below we show the command to use `analyze.trade.costs` to generate a text summary in `R` of the trading, static descriptive, and dynamic descriptive data included in this package:

```
> analyze.trade.costs(trade.mar.2007, dynamic.mar.2007, static.mar.2007, out.fmt = "text")

Trade Cost Analysis


Summary Statistics
```

```
 Total Market Value:        54,109,328
 Total Percent Cost:                0.08
 Max Percent Cost:                  3.72
 Min Percent Cost:                 -4.03
 First Period:              2007-03-01
 Last Period:               2007-03-30
```

Best and worst ID's over all periods

	id	symbol	exec.qty	exec.mkt.val	percent.cost	cost
1	04743910	ACAP	17,800	56,883	-1.20	-682
2	75007710	IMAX	11,600	204,775	-0.82	-1,685
3	83215410	KEYN	18,600	286,637	-0.72	-2,077
4	45245E10	DCO	17,600	90,295	-0.63	-566
5	76073710	ISSC	16,300	191,043	-0.56	-1,069
261	98157910	NANO	101,734	414,787	0.78	3,227
262	39152310	DVW	19,100	255,806	0.89	2,280
263	00912830	AIXD	400	9,539	1.58	151
264	68658810	WOLF	26,400	256,069	2.65	6,788
265	05714910	BUD	2,400	58,270	3.72	2,170

Best and worst periods over all ID's

	period	exec.qty	exec.mkt.val	percent.cost	cost
1	2007-03-01	319,200	1,605,346	-0.42	-6,669
2	2007-03-28	119,401	971,911	-0.08	-750
3	2007-03-22	354,361	4,117,428	-0.05	-2,183
4	2007-03-13	374,622	3,651,345	0.05	1,758
5	2007-03-02	741,100	5,872,369	0.06	3,280
12	2007-03-21	221,489	2,838,467	0.10	2,718
13	2007-03-23	501,588	4,682,846	0.13	6,285
14	2007-03-05	822,195	6,756,640	0.14	9,336
15	2007-03-15	296,574	3,368,101	0.33	11,081
16	2007-03-29	56,322	396,060	1.11	4,388

Best and Worst trades

	id	symbol	period	side	exec.qty	exec.mkt.val	percent.cost	cost
1	31659U30	FICC	2007-03-13	S	1900	4593	-4.0	-185
2	03818830	ADSX	2007-03-01	X	60600	97142	-1.7	-1615
3	88362320	KOOL	2007-03-01	X	31400	94385	-1.3	-1235
4	04743910	AGIX	2007-03-19	C	17800	56883	-1.2	-682
5	75007710	RACK	2007-03-22	C	9000	157736	-1.1	-1694
430	00912830	AIRM	2007-03-09	S	400	9539	1.6	151
431	63007710	NANO	2007-03-29	X	13800	98128	2.6	2574
432	68658810	OHB	2007-03-15	X	26400	256069	2.7	6788
433	98157910	WRSP	2007-03-29	X	12500	47244	2.8	1309
434	05714910	BKR	2007-03-05	B	2400	58270	3.7	2170

NA REPORT

	count
side	8
base.price	0
benchmark.price	2
exec.qty	0

Notice that in addition to inputting the three data frames of raw data, the parameter `out.fmt = "text"` specifies that we want a text summary to be generated. The text output above generates five distinct tables in the summary output. As you can see above, the first table includes summary information of the trade costs analysis for the entire data set. The second, third, and fourth tables show the best and worst trades grouped by `id`, time period, and individual trades. The final table is a report on the `Ca's` found in the data that might be of interest to the user.

In addition, we can run the same function without inputting static descriptive data. This is useful in instances where alternative identifications such as ticker symbols are not needed or not readily available. We simply run the command without `static.mar.2007`:

```
> analyze.trade.costs(trade.mar.2007, dynamic.mar.2007, out.fmt = "text")

Trade Cost Analysis


Summary Statistics
 Total Market Value:        54,109,328
 Total Percent Cost:                 0.08
 Max Percent Cost:                   3.72
 Min Percent Cost:                  -4.03
 First Period:              2007-03-01
 Last Period:               2007-03-30


Best and worst ID's over all periods
          id exec.qty exec.mkt.val percent.cost   cost
1   04743910   17,800       56,883        -1.20   -682
2   75007710   11,600      204,775        -0.82 -1,685
3   83215410   18,600      286,637        -0.72 -2,077
4   45245E10   17,600       90,295        -0.63   -566
5   76073710   16,300      191,043        -0.56 -1,069
261 98157910  101,734      414,787         0.78  3,227
262 39152310   19,100      255,806         0.89  2,280
263 00912830      400        9,539         1.58    151
264 68658810   26,400      256,069         2.65  6,788
265 05714910    2,400       58,270         3.72  2,170


Best and worst periods over all ID's
       period exec.qty exec.mkt.val percent.cost   cost
1  2007-03-01  319,200    1,605,346        -0.42 -6,669
2  2007-03-28  119,401      971,911        -0.08   -750
3  2007-03-22  354,361    4,117,428        -0.05 -2,183
4  2007-03-13  374,622    3,651,345         0.05  1,758
5  2007-03-02  741,100    5,872,369         0.06  3,280
12 2007-03-21  221,489    2,838,467         0.10  2,718
13 2007-03-23  501,588    4,682,846         0.13  6,285
14 2007-03-05  822,195    6,756,640         0.14  9,336
15 2007-03-15  296,574    3,368,101         0.33 11,081
16 2007-03-29   56,322      396,060         1.11  4,388


Best and Worst trades
          id     period side exec.qty exec.mkt.val percent.cost  cost
1   31659U30 2007-03-13    S     1900         4593         -4.0  -185
2   03818830 2007-03-01    X    60600        97142         -1.7 -1615
```

```
3   88362320 2007-03-01    X    31400        94385         -1.3 -1235
4   04743910 2007-03-19    C    17800        56883         -1.2  -682
5   75007710 2007-03-22    C     9000       157736         -1.1 -1694
430 00912830 2007-03-09    S      400         9539          1.6   151
431 63007710 2007-03-29    X    13800        98128          2.6  2574
432 68658810 2007-03-15    X    26400       256069          2.7  6788
433 98157910 2007-03-29    X    12500        47244          2.8  1309
434 05714910 2007-03-05    B     2400        58270          3.7  2170


NA REPORT
                count
side                8
base.price          0
benchmark.price     2
exec.qty            0
```

As you can see above the same tables are generated with the exception that there is no field for symbol in the tables.

PDF reports are also easily generated using the `analyze.trade.costs` function. The reports contain the same tables and information as the text summaries in a convenient PDF format. The reports are available in both normal and verbose versions and can be generated by simply changing the `out.fmt` parameter. Also, reports can simply be displayed or displayed and saved to a file path specified by the parameter `pdf.file`. For example, the command:

```
analyze.trade.costs(trade.mar.2007, dynamic.mar.2007, static.mar.2007, out.fmt = "pdf")
```

would generate and display a normal PDF report of the raw data included in the package while the following line of code:

```
analyze.trade.costs(trade.mar.2007, dynamic.mar.2007, static.mar.2007, out.fmt = "pdf-verbose")
```

would generate and display a verbose version of the report *and* save the PDF report to the file path specified by `pdf.file`. Again, as with the text summaries, static descriptive data is still an optional parameter. Section 4.1 will explain in more detail the usage of `analyze.trade.costs` and the requirements for the raw data input.

4 Detailed overview of tradeCosts package

This section provides overviews of the key components of the `tradeCosts` package. Specifically, it offers detailed descriptions of the function `analyze.trade.costs` and classes `tradeCosts` and `tradeCostsResults`.

4.1 User Interface: `analyze.trade.costs`

To simplify usage of this package a user-level function in the package, `analyze.trade.costs`, was developed. `analyze.trade.costs` allows users to input their raw data and specify if they wish to simply display a summary of their trade costs in the `R` environment and/or generate a verbose or normal PDF report on their trade costs data. The parameters are shown in the function signature below:

```
function (trade, dynamic, static = NULL, benchmark.price = "vwap",
num.trades = 5, analysis.title = "Trade Cost Analysis", out.fmt = "pdf",
pdf.file = NULL)
```

The parameters are explained in more detail below:

- `trade` A `data.frame` of trading data with the following columns required:
  - `id`. the ID of the security being traded.
  - `period`. the time period when the trade was executed.
  - `side`. the`analyze.trades.costs` side of the trade made. Sides should be indicated by the characters `B`,`S`, `X`, and `C` for buy, sell, short, and cover respectively.
  - `exec.price`. the execution price of the trade.
- `dynamic.` A `data.frame` of dynamic data on securities with the following columns required:
  - `id`. the ID of the security being traded.
  - `period`. the time period when the trade was executed.
  - `vwap`. the volume weighted average price (VWAP).
- `static.` A `data.frame` of static data on securities. By default this slot is `NULL`. Although the user can specify a `data.frame` for this slot, it is not necessary for the user to supply static descriptive data. At this stage of the `tradeCosts` package's development, the only data that would be used from `static.desc` is `symbol`, the symbol of the security being traded (an alternate identification from ID).
- `benchmark.price.` A `character` specifying the name of the column of the benchmark price.
- `num.trades.` A number specifying the top `num.trades` best and worst trades, periods, and securities to be displayed in the text summary and/or PDF reports. If num.trades is 1, the best and worst trade will be displayed for each category.
- `analysis.title.` The `character` name of the analysis. By default it is set as `"Trade Cost Analysis 1.0"`.
- `out.fmt.` A `character` specifying the report type to be generated. Input `"pdf"` for a normal PDF report, `"pdf-verbose"` for a verbose PDF report, or `"text"` for a text summary of the data displayed in the `R` console.
- `pdf.file.` A `character` path to the file you want the PDF report to be saved too. If a PDF report was not selected as the output type then this argument is ignored. By default, `pdf.file` is set as `NULL` - this causes the PDF report to be saved in a temporary directory in `R`. This temporary directory is also where the all the compiling of `Sweave` files and LaTeX files are done.

Function `analyze.trades.costs` works by creating a `tradeCosts` and `tradeCostsResults` object from the input data. The two classes are explained in more detail in the section 4.2. `analyze.trades.costs` generates its PDF reports by using package `Sweave` to generate `tex` files to be compiled from template files found in `/inst/template/`. Detailed examples of the use of this function were given above in section 3.

4.2 `tradeCosts` Class

The basic structure of the `tradeCosts` package consists of two classes, `tradeCosts` and `tradeCostsResults`. Class `tradeCosts` has slots to take in the data required for the trade costs analysis. To begin an analysis of trade costs, an object of `tradeCosts` class with the necessary data in its slots is created. The slots are listed and explained in detail below:

- `name`. A string of the name of this trade costs analysis.
- `trade.data`. A `data.frame` of trade data with the following columns required:
  - `id`. the ID of the security being traded.

  - `period`. the time period when the trade was executed.
  - `side`. the`analyze.trades.costs` side of the trade made. Sides should be indicated by the characters `'B'`,`'S'`, `'X'`, and `'C'` for buy, sell, short, and cover respectively.
  - `exec.price`. the execution price of the trade.

- `dynamic.desc`. A `data.frame` of dynamic data on securities with the following columns required:
  - `id`. the ID of the security being traded.
  - `period`. the time period when the trade was executed.
  - `vwap`. the volume weighted average price (VWAP).

- `static.desc`. A `data.frame` of static data on securities.Although the user can specify a `data.frame` for this slot, data from `static.desc` is not necessary for analysis to proceed.

The `tradeCosts` also has a method, `analyzeData`, which merges the `trade.data`, `dynamic.desc`, and `static.desc` data into one data set, counts the number of `NA`s in the merged data set, calculates cost and percent cost, and removes extraneous columns. After performing these operations `mergeData` passes its results into a new object of class `tradeCostsResults` which is discussed above in section 3.

4.3 `tradeCostsResults` Class

Class `tradeCostsResults` stores a single `data.frame` object containing all the stripped down trade data with cost information. The class also contains methods for calculating summary statistics and generating the text summary and PDF reports. Its slots are listed and explained in detail below:

- `name.` A string of the name of this trade costs analysis.
- `results.` A `data.frame` of merged and pared down raw data.
- `na.counts.` A `data.frame` of counts made of NAs in the raw data

Class `tradeCostsResults` has two methods which produce end output for the user, `summary` and `pdfsummary`. Method `summary` calculates and displays a text summary report in the `R` console. Method `pdfsummary` generates the summary PDF reports, both normal and verbose.

5 Conclusion

For those who engage in frequent transactions in financial markets, trade costs can play a pivotal role in portfolio performance. Institutions and individuals who wish to improve their returns can do so by understanding and carefully managing the costs incurred during trading. This package provides a set of tools for generating summaries and reports on a component of trade costs, slippage, from raw trading data.

Jeff Enos, David Kane, Aaron Schwartz, and Luyi Zhao
Kane Capital Management
Cambridge, Massachusetts, USA
`jeff@kanecap.com`, `david@kanecap.com`,
`aaron.j.schwartz@williams.edu` and `luyizhao@fas.harvard.edu`

References

[1] Robert Kiseel and Morton Glantz. *Optimal Trading Strategies*. American Management Association, 2003.