Part 5: Visualization

Summary

This Part covers tools for visualizing data in R.

Powerful visualization techniques are critical to discovering and understanding the relationships between variables.

Example Data Set

First, we will revisit the data set we considered in the previous Part, and the steps needed to properly format the data:

```
> fulldata = read.table("q1_2017_all.csv", sep=",", header=T, quote="")
> fulldata$Date = as.Date(as.character(fulldata$Date), format="%Y%m%d")
> fulldata = fulldata[!duplicated(fulldata[,c(1,3)], fromLast=TRUE),]
> fulldata$McapRank = factor(fulldata$McapRank, ordered=TRUE)
> fulldata$TurnRank = factor(fulldata$TurnRank, ordered=TRUE)
> fulldata$VolatilityRank = factor(fulldata$VolatilityRank, ordered=TRUE)
> fulldata$PriceRank = factor(fulldata$PriceRank, ordered=TRUE)
```

The entire data set is a bit large to work with conveniently, so we are going to restrict consideration to 1000 equities:

```
> tickkeep = unique(fulldata$Ticker)[seq(1,
+       length(unique(fulldata$Ticker)),length=100)]
> fulldata = fulldata[fulldata$Ticker %in% tickkeep,]
```

Classic Plotting Functions in R

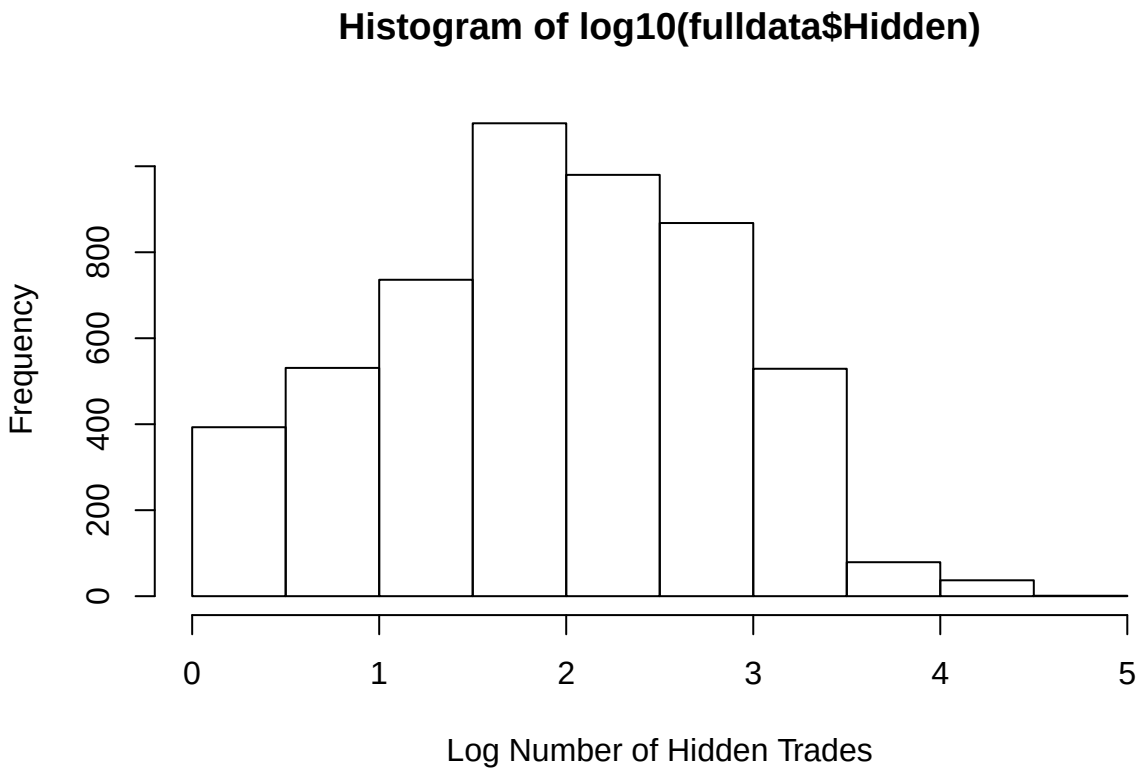
Prior to moving on the more advanced visualization functions, it is important to understand the basic plotting capabilities of R.

We saw the use of `boxplot()` in a previous Part. A similar view of the shape of a distribution can be obtained using a histogram, formed using the function `hist()`:

```
> hist(log10(fulldata$Hidden),
+      xlab="Log Number of Hidden Trades")
```

The title on the plot can be changed using the `main` argument.

The number of bins in the histogram can be altered using the `breaks` argument.

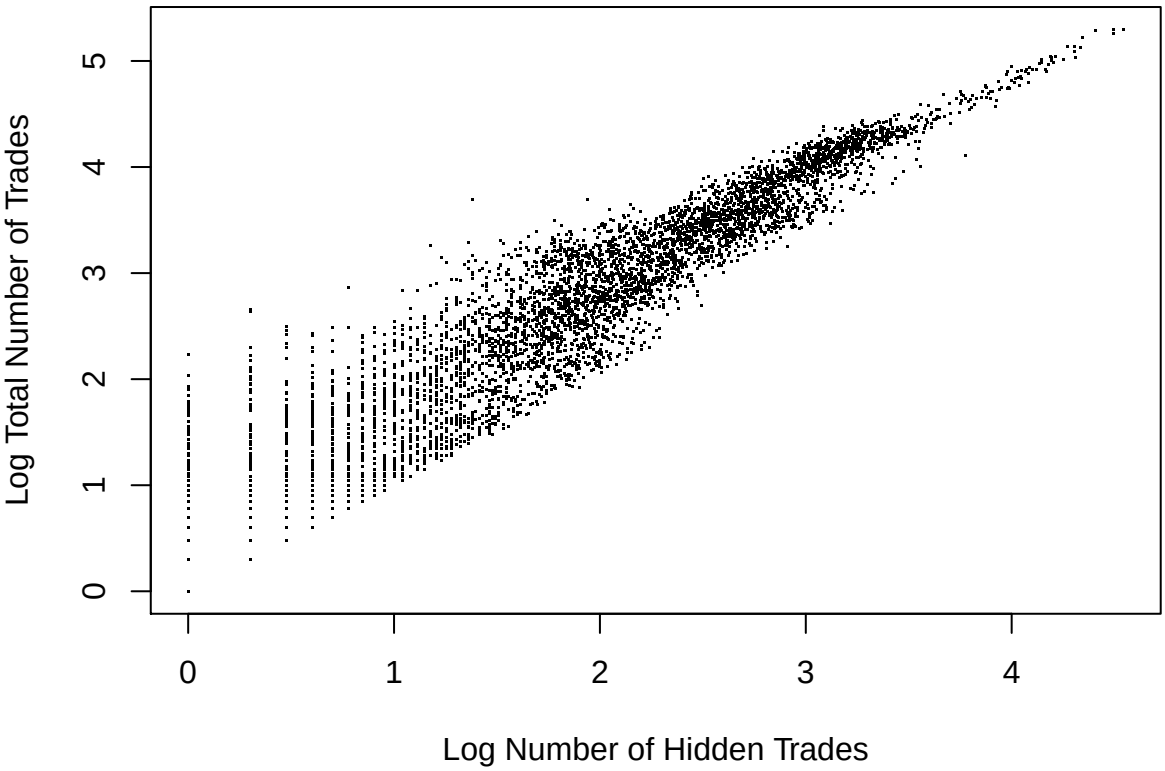

Histogram of log10(fulldata$Hidden)
Frequency
0
200
400
600
800
0
1
2
3
4
5
Log Number of Hidden Trades

Scatter plots are ubiquitous, as well. They are created simply in R using the `plot()` function:

```
> plot(log10(fulldata$Hidden),
+      log10(fulldata$TradesForHidden),
+      xlab="Log Number of Hidden Trades",
+      ylab="Log Total Number of Trades", pch=".")
```

The argument `pch` changes the plotting character. In a case such as this, with so many dots on the plot, the default plotting character is too large.

Many other properties of the scatter plot can be adjusted using the `par()` function. See `help(par)` for more information.


Log Total Number of Trades
0
1
2
3
4
5
0
1
2
3
4
Log Number of Hidden Trades

1

Introduction to ggplot2

The package `ggplot2` contains a great deal of valuable data visualization tools. This package can be installed, along with other useful packages for working with large and complex data sets, via the `tidyverse` meta-package:

```
> install.packages("tidyverse")
```

The collection of packages in tidyverse comprise “a collection of R packages that share common philosophies and are designed to work together." – tidyverse.org

The `ggplot2` package implements a grammar of graphics, where plots are created by combining geoms.

The syntax is a little awkward at first, but this approach yields a great deal of flexibility, and a great deal of power in exploring features in a data set.

A plot is initialized by using the `ggplot()` function:

```
> ggplot(data = fulldata)
```

To the basic plot, one adds the "aesthetic" via the `mapping` option. An aesthetic a generic concept for the visual features of a plot. In this case, we are specifying that the variable on the horizontal axis is `Hidden`, and the variable on the vertical axis is `Cancels`:

```
> ggplot(data = fulldata, mapping =
+                  aes(x=Hidden, y=Cancels))
```

We can now add onto this base to create plots with complex features. You can, for instance, define

```
> baseplot = ggplot(data = fulldata, mapping =
+                  aes(x=Hidden, y=Cancels))
```

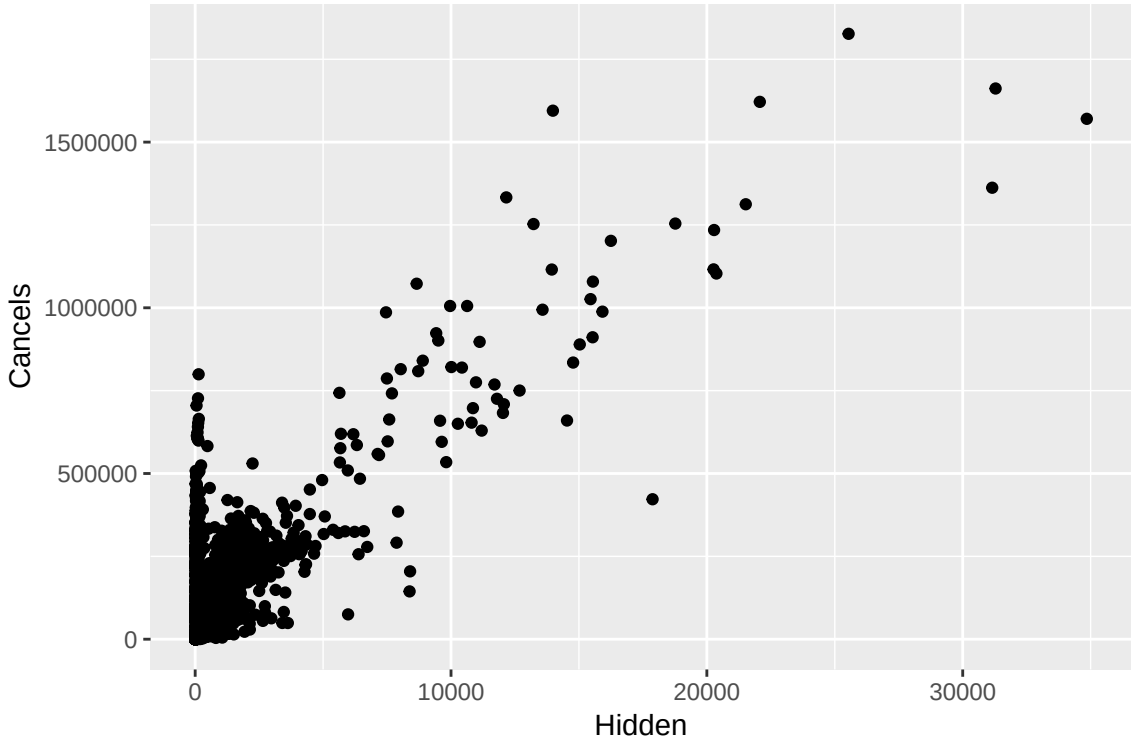
and add onto `baseplot` in what follows.

A Simple Scatter Plot

Consider the following syntax:

```
> baseplot + geom_point()
```

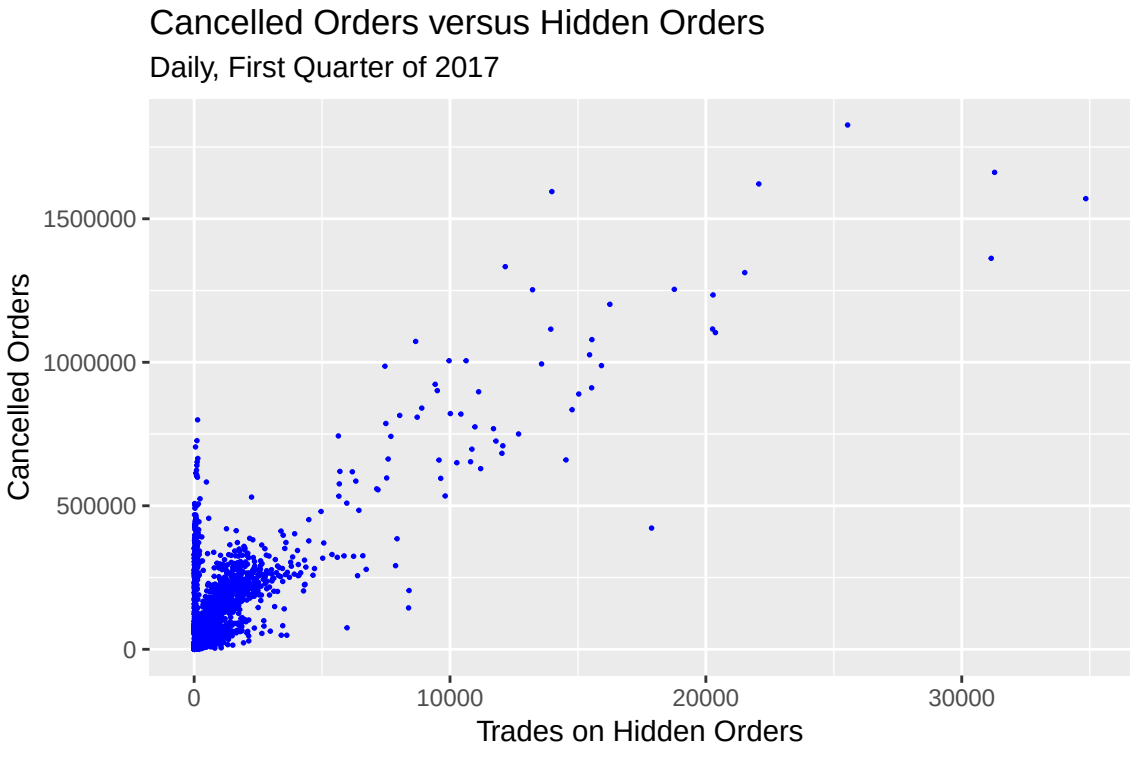
Note how the `geom_point()` function is used to add a scatter plot onto the current figure, as stored in `baseplot`.


1500000
1000000
500000
0
Cancels
0
10000
20000
30000
Hidden

1

1 Now we will work to improve the appearance of this plot.

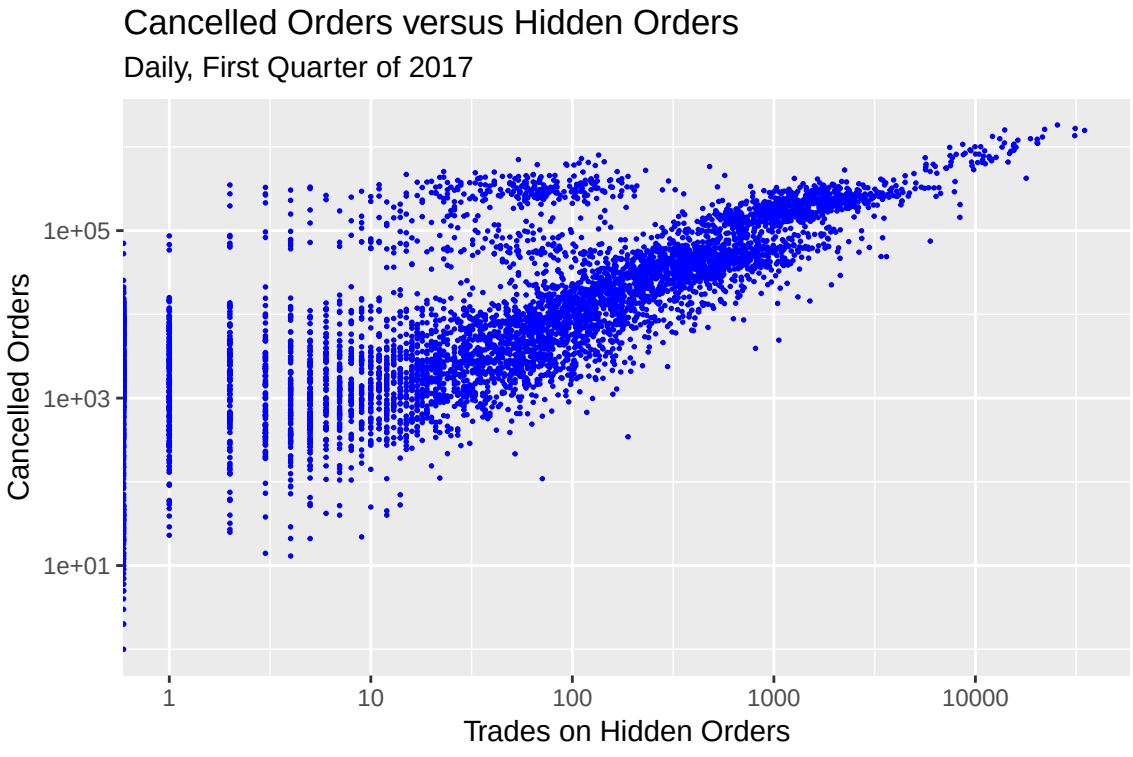
```
> baseplot +
+   geom_point(size=0.3, color="blue") +
+   labs(x="Trades on Hidden Orders",
+        y="Cancelled Orders",
+        title="Cancelled Orders versus Hidden Orders",
+        subtitle="Daily, First Quarter of 2017")
```


Cancelled Orders versus Hidden Orders
Daily, First Quarter of 2017
Cancelled Orders
1500000
1000000
500000
0
0
10000
20000
30000
Trades on Hidden Orders

Exercise: Describe the changes that were made to improve the plot.

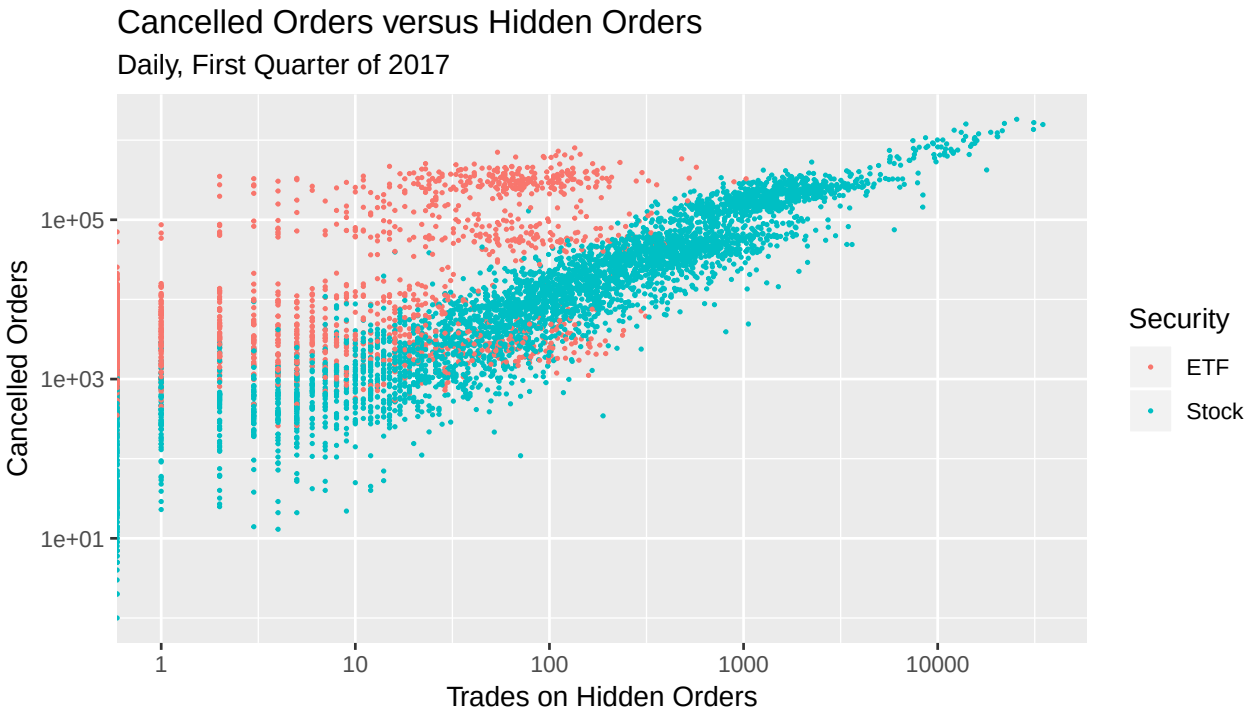
1 Next, we will convert the axes to the log scale:

```
> baseplot +
+   geom_point(size=0.3, color="blue") +
+   labs(x="Trades on Hidden Orders",
+      y="Cancelled Orders",
+      title="Cancelled Orders versus Hidden Orders",
+      subtitle="Daily, First Quarter of 2017") +
+   scale_x_log10() + scale_y_log10()
```


Cancelled Orders versus Hidden Orders
Daily, First Quarter of 2017
Cancelled Orders
1e+05
1e+03
1e+01
1
10
100
1000
10000
Trades on Hidden Orders

The `color` argument can be switched to vary with the levels of one of the factors.

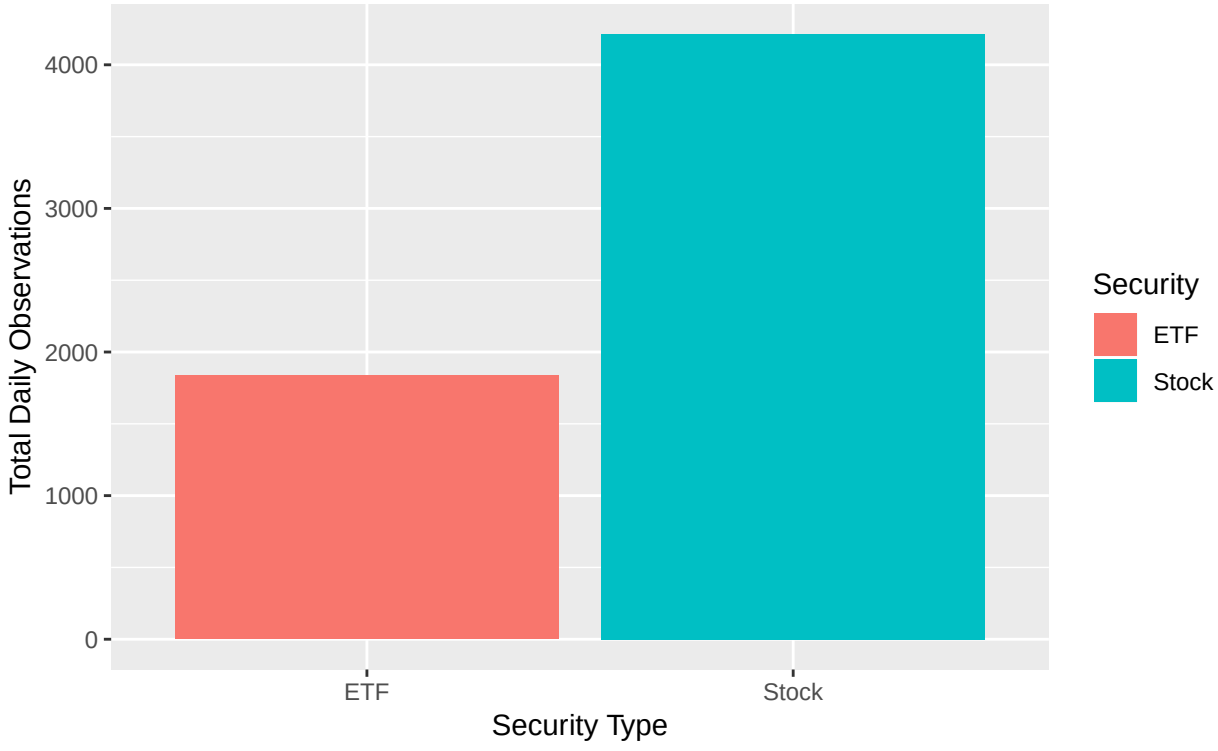
```
> baseplot +
+   geom_point(size=0.3, mapping=aes(color=Security)) +
+   labs(x="Trades on Hidden Orders",
+        y="Cancelled Orders",
+        title="Cancelled Orders versus Hidden Orders",
+        subtitle="Daily, First Quarter of 2017") +
+   scale_x_log10() + scale_y_log10()
```


Cancelled Orders versus Hidden Orders
Daily, First Quarter of 2017
Cancelled Orders
1e+05
1e+03
1e+01
1
10
100
1000
10000
Trades on Hidden Orders
Security
ETF
Stock

Bar Plots

Basic bar charts can be created to inspect the distribution of a factor, using the `geom_bar()` function:

```
> ggplot(data=fulldata,
+      mapping=aes(x=Security, fill=Security)) +
+    geom_bar() +
+    labs(x="Security Type",
+         y="Total Daily Observations")
```


4000
3000
2000
1000
0
Total Daily Observations
ETF
Stock
Security Type
Security
ETF
Stock

1

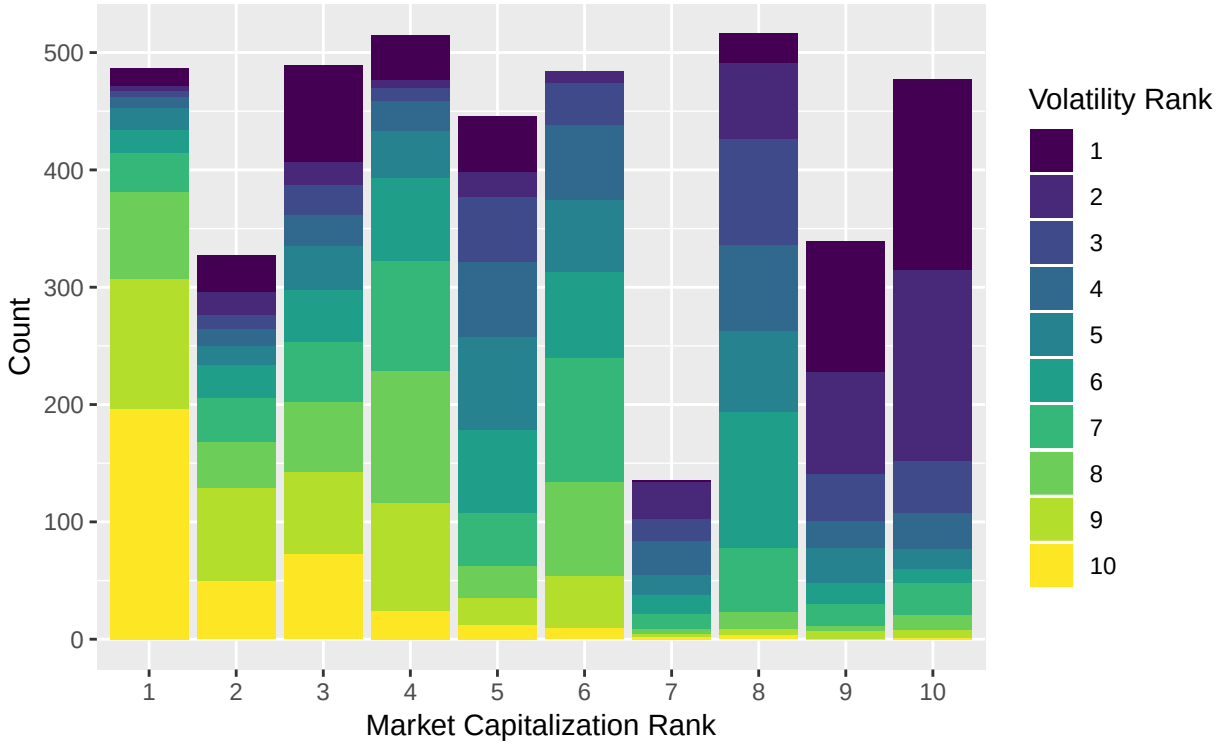
Exercise: Now that we have seen a few examples, try to explain the notion of the "aesthetic."

For the next plot, we will restrict to data that do not have missing values on `McapRank`, and are of type "Stock":

```
> fulldataStock = filter(fulldata,
+         Security == "Stock" & !is.na(McapRank))
```

We will adjust the aesthetic so that the "fill" color of the bars corresponds to the level of `VolatilityRank`:

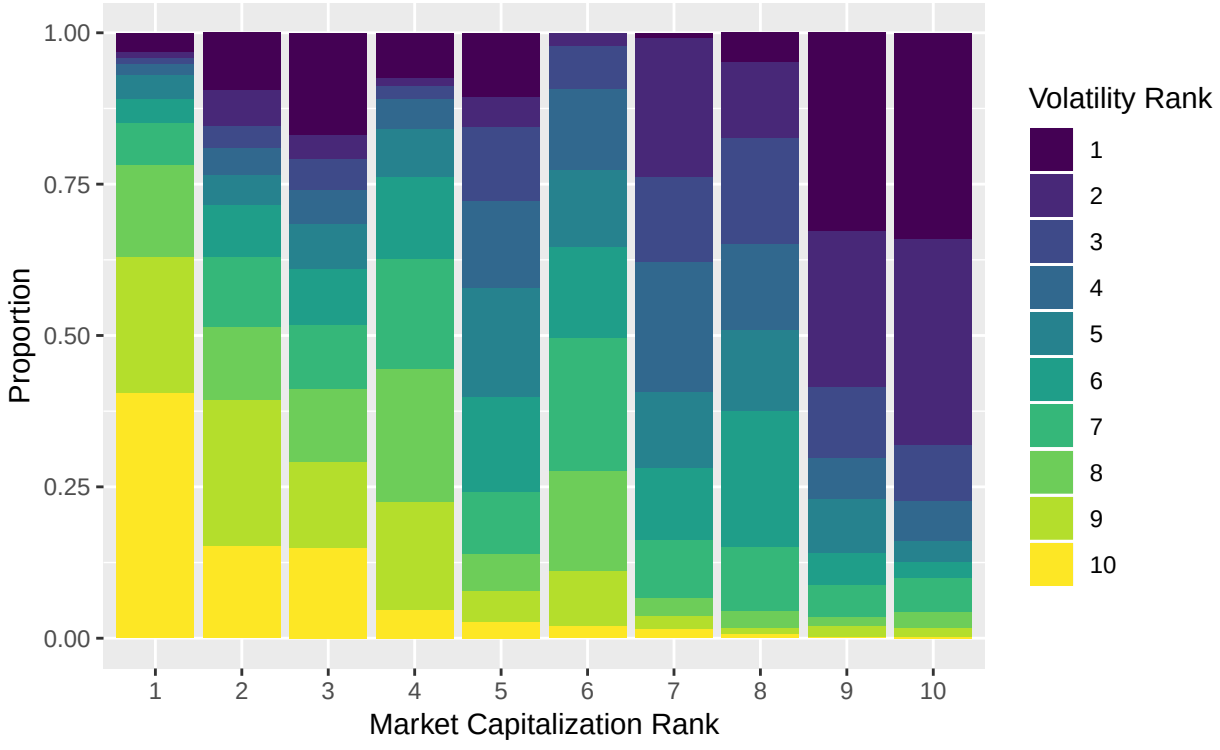
```
> ggplot(data=fulldataStock,
+     mapping=aes(x=McapRank, fill=VolatilityRank)) +
+   geom_bar() +
+   labs(x="Market Capitalization Rank", y="Count",
+     fill="Volatility Rank")
```


Count
500
400
300
200
100
0
1
2
3
4
5
6
7
8
9
10
Market Capitalization Rank
Volatility Rank
1
2
3
4
5
6
7
8
9
10

The bars in the previous plot are not of equal height because of the sub-sampling we did to reduce to 1000 equities.

We can change vertical axis to proportion by setting `position="fill"`:

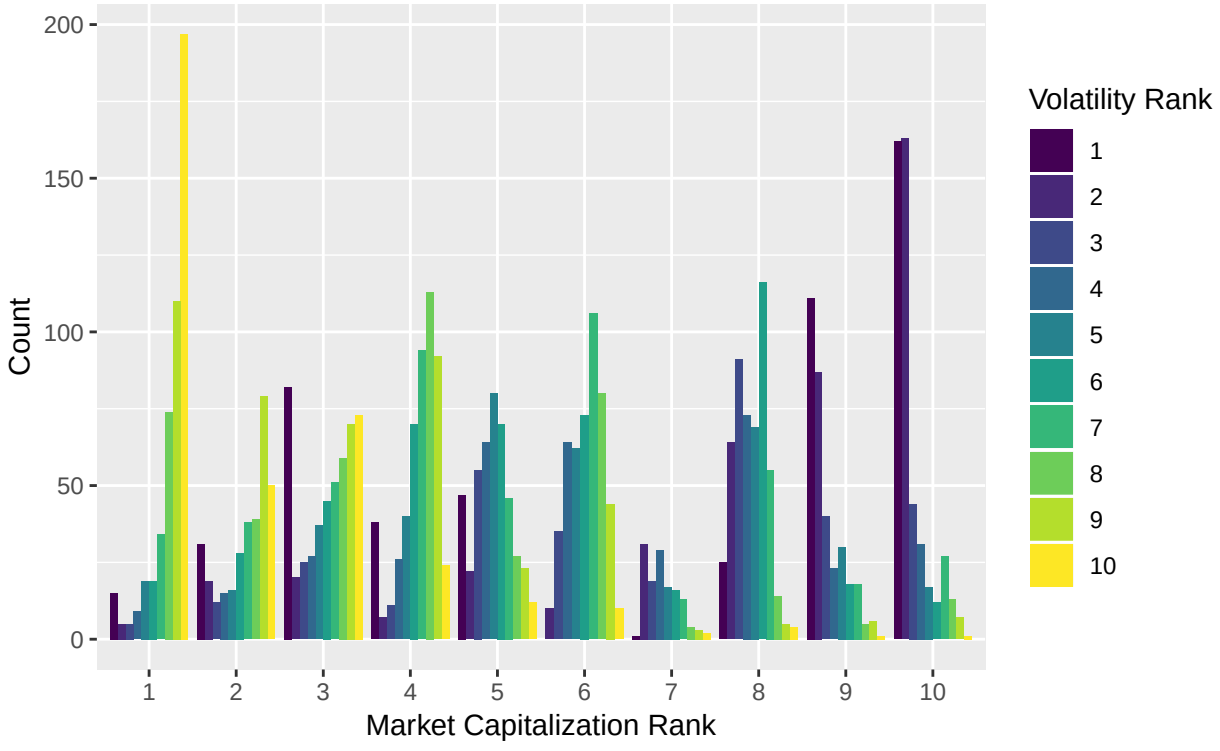
```
> ggplot(data=fulldataStock,
+        mapping=aes(x=McapRank, fill=VolatilityRank)) +
+   geom_bar(position="fill") +
+   labs(x="Market Capitalization Rank",
+        y="Proportion", fill="Volatility Rank")
```


1.00
0.75
0.50
0.25
0.00
Proportion
1
2
3
4
5
6
7
8
9
10
Market Capitalization Rank
Volatility Rank
1
2
3
4
5
6
7
8
9
10

1

By setting `position="dodge"`, the bars are arranged more like a histogram:

```
> ggplot(data=fulldataStock,
+      mapping=aes(x=McapRank, fill=VolatilityRank)) +
+    geom_bar(position="dodge") +
+    labs(x="Market Capitalization Rank",
+      y="Count", fill="Volatility Rank")
```


Count
200
150
100
50
0
1
2
3
4
5
6
7
8
9
10
Market Capitalization Rank
Volatility Rank
1
2
3
4
5
6
7
8
9
10

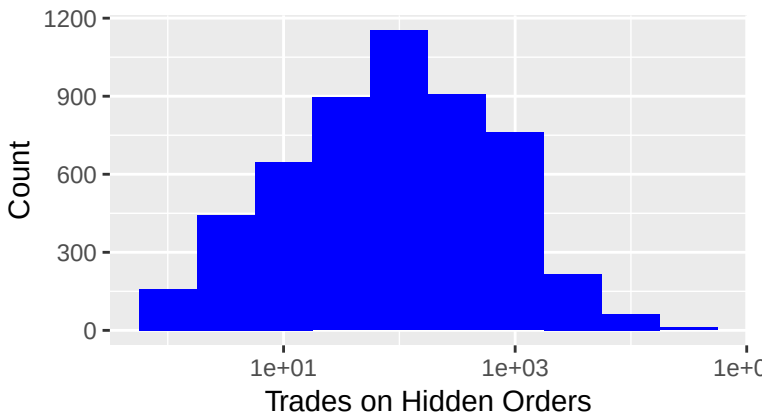
1

Exercise: Why would you not want to create a scatter plot to explore the relationship between Market Capitalization Rank and Volatility Rank? Consider how the function `geom_jitter()` can help in this regard.

1 ## Histograms

2 The function `geom_histogram()` will create a histogram:

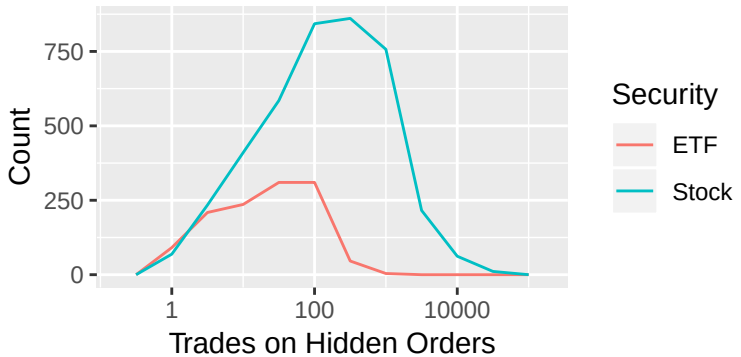
```
> ggplot(data=fulldata, mapping=aes(x=Hidden)) +
+   geom_histogram(binwidth=0.5, fill="blue") +
+   scale_x_log10() +
+   labs(x="Trades on Hidden Orders", y="Count")
```



3

Overlapping histograms can be difficult to interpret (try it!). An alternative is to create frequency polygons using `geom_freqpoly()`:

```
> ggplot(data=fulldata,
+        mapping=aes(x=Hidden, color=Security)) +
+   geom_freqpoly(binwidth=0.5, fill="blue") +
+   scale_x_log10() +
+   labs(x="Trades on Hidden Orders", y="Count")
```

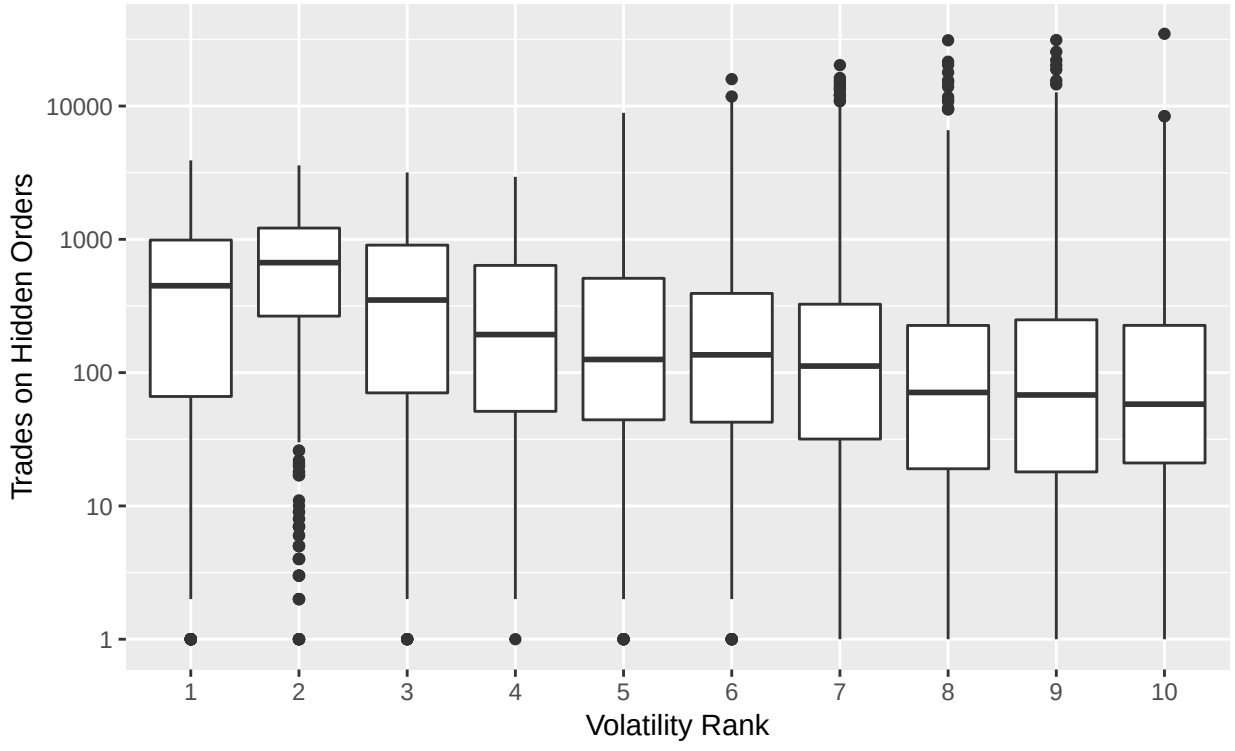


Exercise: Both `geom_histogram()` and `geom_freqpoly()` have an argument `binwidth`. Discuss the technical and practical effects of varying `binwidth`.

Boxplots

Side-by-side boxplots are also useful for comparing the distributions of variables over different values of a factor. For example,

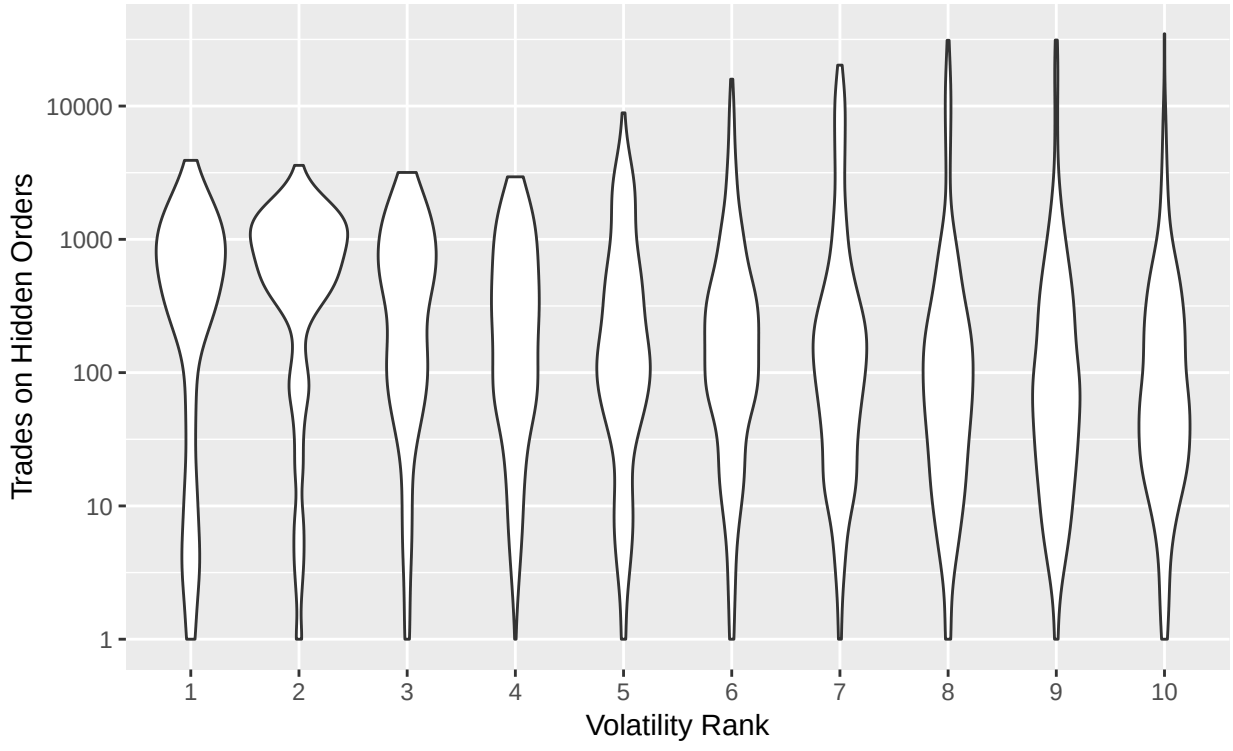
```
> ggplot(data=fulldataStock,
+        mapping=aes(x=VolatilityRank, y=Hidden)) +
+   geom_boxplot() + scale_y_log10() +
+   labs(y="Trades on Hidden Orders",
+        x="Volatility Rank")
```


Trades on Hidden Orders
10000
1000
100
10
1
1
2
3
4
5
6
7
8
9
10
Volatility Rank

1

A violin plot shows the same type of information, with a little more detail. The width of a "violins" is greater in areas where a greater proportion of the observations lie.

```
> ggplot(data=fulldataStock,
+         mapping=aes(x=VolatilityRank, y=Hidden)) +
+   geom_violin() + scale_y_log10() +
+   labs(y="Trades on Hidden Orders",
+        x="Volatility Rank")
```

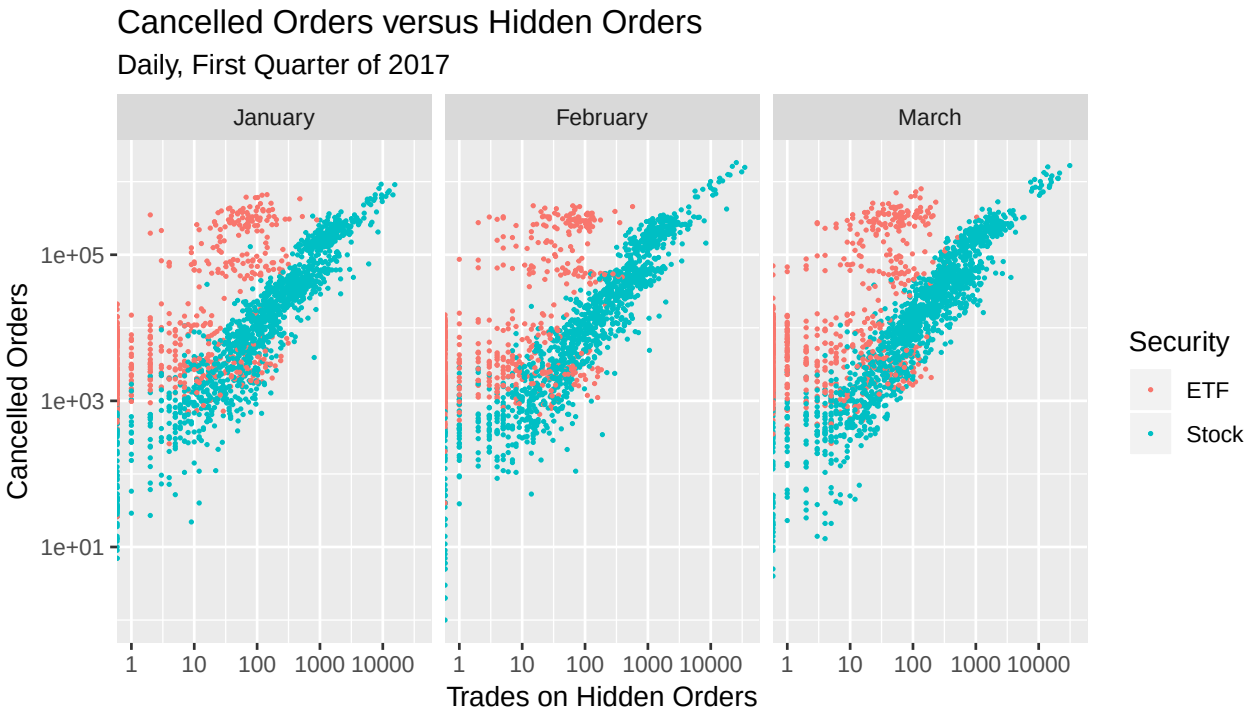

Trades on Hidden Orders
10000
1000
100
10
1
1
2
3
4
5
6
7
8
9
10
Volatility Rank

1

Facets

Facets refer to an arrangment of plots on which one or more factors vary. For example, return to our scatter plot example above, and use the `facet_grid()` function to break up the plots by month:

```
> baseplot +
+   geom_point(size=0.3, mapping=aes(color=Security)) +
+   labs(x="Trades on Hidden Orders",
+        y="Cancelled Orders",
+        title="Cancelled Orders versus Hidden Orders",
+        subtitle="Daily, First Quarter of 2017") +
+   scale_x_log10() + scale_y_log10() +
+   facet_grid(.~factor(months(Date),
+         levels=c("January","February","March")))
```


Cancelled Orders versus Hidden Orders
Daily, First Quarter of 2017
January
February
March
Cancelled Orders
1e+05
1e+03
1e+01
1
10
100
1000
10000
Trades on Hidden Orders
Security
ETF
Stock

Exercise: What happens in the previous example if the command

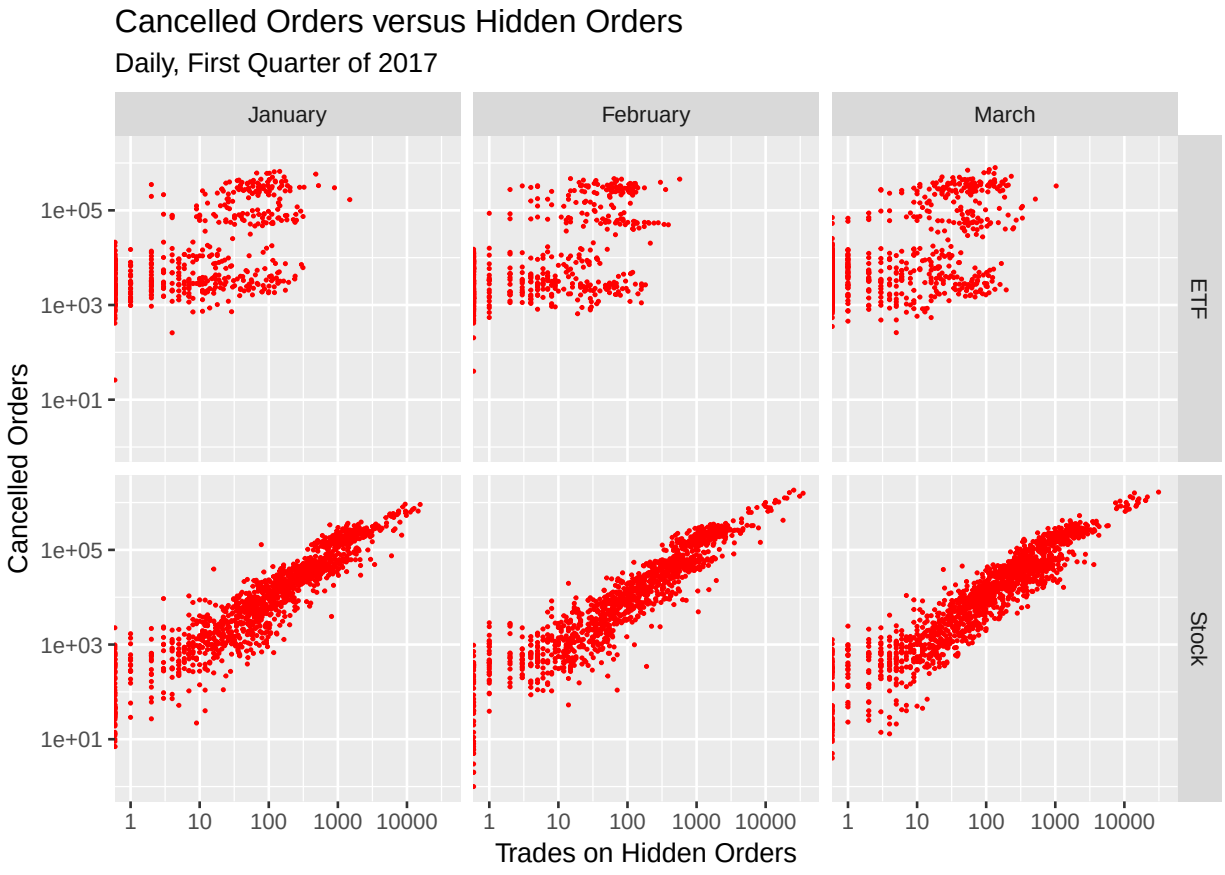
```
> facet_grid(.~months(Date))
```

is used instead? What about if use the following?

```
> facet_grid(months(Date)~.)
```

1 Two factors can vary:

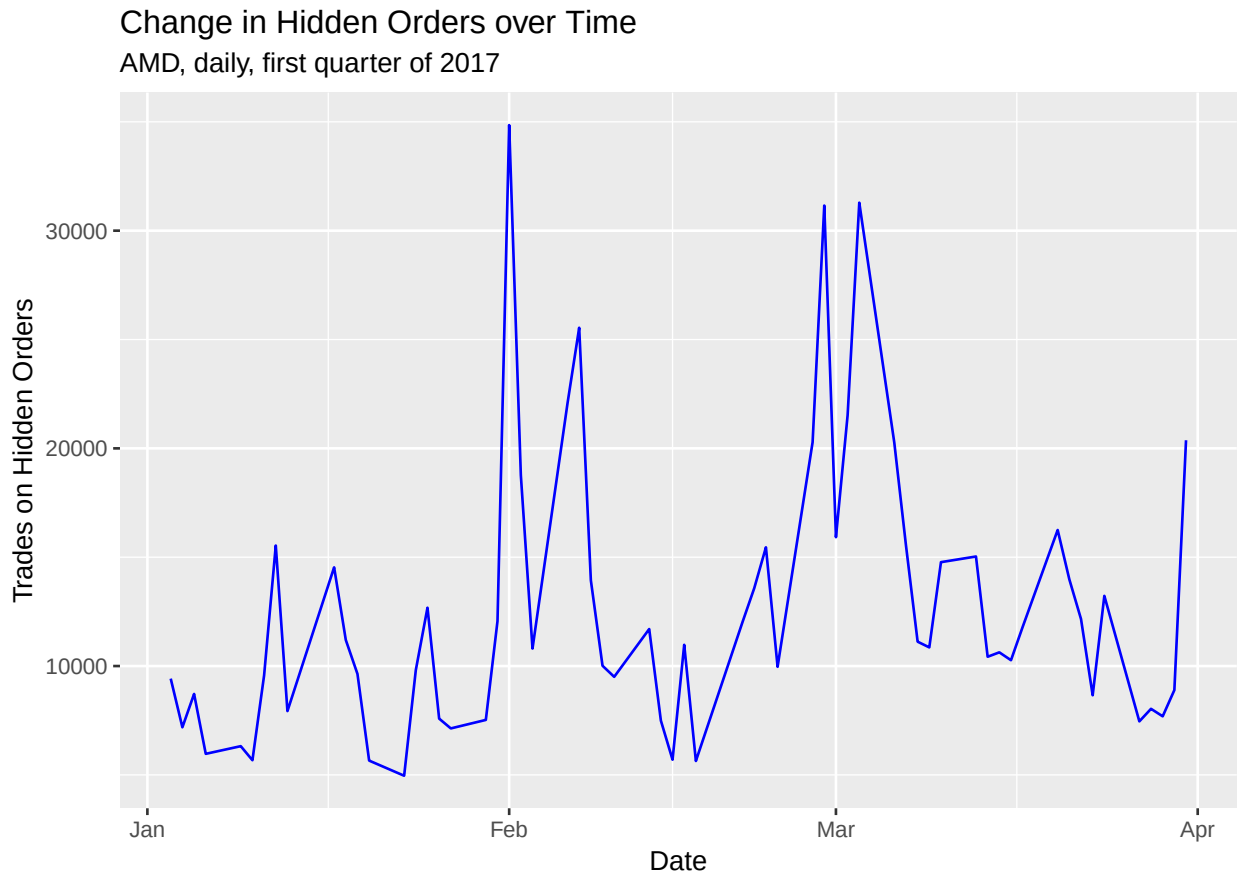
```
> baseplot +
+  geom_point(size=0.3, color="red") +
+  labs(x="Trades on Hidden Orders",
+       y="Cancelled Orders",
+       title="Cancelled Orders versus Hidden Orders",
+       subtitle="Daily, First Quarter of 2017") +
+  scale_x_log10() + scale_y_log10() +
+  facet_grid(Security~factor(months(Date),
+        levels=c("January","February","March")))
```


Cancelled Orders versus Hidden Orders
Daily, First Quarter of 2017
January
February
March
ETF
Stock
Cancelled Orders
1e+05
1e+03
1e+01
1
10
100
1000
10000
Trades on Hidden Orders

Time Series Plots

We will, of course, be working with a great deal of time series data. The `geom_line()` function is useful:

```
> ggplot(data=filter(fulldata,Ticker=="AMD"),
+        mapping=aes(x=Date,y=Hidden)) +
+   geom_line(color="blue") +
+   labs(x="Date", y="Trades on Hidden Orders",
+      title="Change in Hidden Orders over Time",
+      subtitle="AMD, daily, first quarter of 2017")
```


Change in Hidden Orders over Time
AMD, daily, first quarter of 2017
Trades on Hidden Orders
30000
20000
10000
Jan
Feb
Mar
Apr
Date