Portfolio123 Virtual Strategy Design Class
By Marc Gerstein

Supplement 2 – Cost of Capital; (2) A Usable Approach

In WACC - 1, we defined Cost of Capital, or WACC, and discussed some serious practical problems implementing it. In this installment, we're going to take a first stab at calculating a usable WACC.

We'll follow the basic formula: (WtD*CostD)+ (WtP*CostP)+(WtC*CostC) where,

WtD = Weight (%) of Debt
CostD Cost (%) of Debt
WtP = Weight (%) of Preferred Equity
CostP Cost (%) of Preferred Equity
WtC = Weight (%) of Common Equity
CostC Cost (%) of Common Equity

Warning #1: In order to proceed further, it will be important – critical actually – that you learn to love spit and chewing gum, because I'm going to use a heck of a lot of both. But that's OK. You might as well get used to it if you aren't already. Consider what we do; we use data from the past to help us make investment decisions that will succeed or fail based on what happens in the future. (We all know what they say about past performance not assuring future outcomes; a now-boilerplate adage that just so happens to be true.) So in a sense, EVERYTHNG we do, no matter how carefully we think we worked, is really put together with spit and chewing gum. It's a fundamental law of investing. And we might as well call it Gerstein's Law since I can't think of anybody else who'd actually want to take credit for articulating it. Oh well. ☺

Warning #2: This is not the approach I actually use. But considering the tremendous intellectual weight carried by the above formula and by CAPM as a vehicle for computing cost of equity, I think I really need to offer this for the benefit for those who absolutely positively want to use it and to motivate others to consider alternatives such as I'll show starting in Part 3.

We're going to do a basic WACC using CAPM for the common equity component. But we're going to add some rules that identify and repair ridiculous situations caused by the plain-vanilla computations.

Debt Component

Essentially, cost of debt is after-tax interest expense divided by average debt over the course of four quarters. We need four quarters in order to account for seasonal variations. But we're going to supplement this with three conditional rules:

1. We're going to establish an upper boundary as 10 percentage points (1000 basis points) above the risk-free rate. (Obviously, if you want to use the screen I'll present, you can modify that or any other seat-of-the-pants choice.) In other words, if a plain vanilla computation produces a cost of debt of, say, 38%, we'll cut it to the risk-free rate plus 10 percentage points. Believe it or not, we really do need this rule!

2. We're also going to establish a lower boundary that will be equal to 3 percentage points (300 basis points) above the risk-free rate. If the plain vanilla computation produces a result below the boundary, we'll raise it to equal the risk free rate plus three percentage points. And yes, as silly as the need for this seems, we definitely do need it.

3. Finally, if a company reports interest expense but does not report debt (see, e.g., the MSG example referred to in Part 1), we'll set cost of debt at the risk-free rate plus three percentage points.

We're also going to frame several database references in terms of the NA function; we want a NUL data field to be read into the model as zero, not NA. Use of NA would wipe unleveraged companies out of our results.

We're also going to use ShowVar functionality. If you aren't familiar with it, please take some time to get acquainted with it. This is a very powerful part of P123. Although it was intended to strengthen our report-building capabilities, life can throw some interesting curve balls. It turns out that ShowVar actually brings some bona fide programming capabilities into P123.

Here we go:

- Showvar(@Int,ISNA(IntExpTTM*.65,0))
    - *Define @Int as IntExp *.65, which assumes a standard 35% tax rate; if IntExpTTM is NA, then change it to zero*
- Showvar(@Dbt,ISNA((DbtTot(0,qtr)+DbtTot(1,qtr)+DbtTot(2,qtr)+DbtTot(3,qtr)+DbtTot(4,qtr))/5,0))
    - *Define @Dbt and the average of the last five quarterly DbtTot figures; if the result is NA, then change it to zero*
- Showvar (@DbtCost,ISNA((@Int/@Dbt)*100,0))
    - *Do the plain-vanilla cost of debt computation and multiply by 100 to get a % if you like to work that way*
- Showvar (@DbtCost1,Eval(@DbtCost>((close(0,#tnx)/10)+10), (close(,0,#tnx)/10)+10,@DbtCost))
    - *Test for the need to reduce an excessive @DbtCost number to 10 percentage points above the risk-free rate. Depending on how the test fares, let the original @DbtCost number or our newly calculated ceiling figure be a new item, @Dbtcost1*
- Showvar(@DbtCost2,Eval(@DbtCost1<((close(0,#tnx)/10)+3), (close(,0,#tnx)/10)+3,@DbtCost1))
    - Define @DbtCost2 as either the @DbtCost1 item we just calculated or, if that turned out to have been in below the lower threshold, 300 basis points above treasury, then at 300 basis points above treasury
    - Don't worry about seemingly contradicting the previous rule. There is no way the original @DbtCost figure can simultaneously breach both our upper and lower boundaries. Either this rule, or the preceding one, will necessarily always redundant.

- Showvar(@DbtCost3,Eval(@Int=0 and @Dbt!=0,((close(0,#tnx)/10)+3),@DbtCost2))

    - When the screener reaches this rule, we know for sure that we have a tolerable @Dbtcost2 figure. It's equal to the original plain-vanilla @DbtCost item, or @DbtCost1, the upper-boundary figure or @DbtCost2, the lower boundary figure. So we're home on cost of debt . . . almost.

    - Every now and then, the database will still throw us a whacky curve ball, a situation where @Int shows as zero but where @Dbt is non zero. That may be fine based on what goes into financial statement footnotes, but for us, it's intolerable. So in this rule, we test for the whacky phantom-interest condition. If it's present, we define @DbtCost3 as 300 basis points above the risk-free rate. If it's not present, then we simply stick with @DbtCost2 but redefine it, for purposes of clean logic, as @DbtCost3.

@DbtCost3 is the grand finale. That is the figure we'll use as cost of debt in our final calculation of WACC. Different companies will get to @DbtCost3 different ways depending on how clean or messed up their original @DbtCost computations were. Eventually, though, they'll all get there.

I know this seems like a horrifying process. It really isn't. Once you get used to programming with ShowVar, as I strongly suggest you do, you'll see that this is really pretty easy. But even if you really are genuinely repulsed, trust me, you'd be a lot more disgusted if you saw many of the cost of debt numbers you'd be using if you stuck with a the simple after-tax interest divided by average debt computation. (Actually, you don't have to trust me. Investigate and see for yourself.) And besides, given the existence of "copy-and-paste" and "save as," you really only need to do this once.

Preferred Equity Component

Are you ready for an easy solution? Yeah, me too. If we want, we could go through a debt-cost-like rigmarole to compute cost of preferred. But frankly, preferred stock isn't what it used to be in terms of popularity as a financing vehicle. We can easily afford to go whole hog with the spit and chewing gum. We'll just set cost of preferred at a fixed spread above the cost of debt. I'll assume one percentage point; obviously, you can choose a different number.

- SetVar(@PfdCost,@DbtCost3+1)

That's not just spit and chewing gum; that's a huge you-know-what-load of spit and chewing gum! But frankly, there are a ton of things to sweat about when it comes to stock selection and since we all have just so much sweat, we need to prioritize. Let's sweat over things that really matter. This is not one of them.

Common Equity Component

This is a place that warrants more sweat, more spit, and more chewing gum. But having suffered through the horrors of cost of debt (remember the good old days, when you thought cost of debt was easy, like from earlier today before you read any of this), you should actually find this pretty simple. We'll set boundaries around beta just as we did with @DbtCost. But we don't have to use ISNA unless we really want to (there are some NA companies with NA Betas, but I'm filing that under "S" for sweating-the-small-stuff), we don't have to tax-effect anything and we have one less whacky potentiality about which we must worry.

We'll use CAPM but with a boundary-protected beta rather than a database beta.

- ShowVar(@beta,Beta5Y)
    - *We'll start with a standard Beta data-point. If you prefer, you can substitute beta3Y or BetaFunc() for Beta5Y*
- ShowVar(@beta1,Eval(@beta>3,3,@beta))
    - *Now, we set the topmost boundary for Beta. We're going to switch from @beta to @beta1. If @beta is above 3, then we'll set @beta1 to 3. If @beta is below 3, then we'll set @beta1 equal to our original @beta. Obviously, if you want a different ceiling value, 2.5, 4, or whatever, just substitute for 3.*
- ShowVar(@beta2,Eval(@beta1<.7,.7,@beta1))
    - *You know what's coming now, the bottom limit. We'll convert @beta1 to @beta2 by using our if-then capability to determine if @beta1 is below the lower limit we choose, which for me is .7. If @beta 1 is below .7, then I'll set @beta2 at .7. If @beta1 is above .7, then I'll bring the @beta1 value for @beta2.*
- ShowVar (@CAPMRet,(close(0,#TNX)/10)+(@beta2*5))
    - *@beta2, the cleansed version of our starting point, Beta5Y in this example, is what we'll plug into the CAPM cost of equity formula. We're defining the answer as @CAPMRet and calculating it as close(0,#TNX)/10), which is the risk-free rate, plus (@beta2*5), which is the usable beta multiplied by our assumed equity risk premium. Of course you can change the risk premium to any other figure you want. But please do not spend too much time trying to get something more objective. It can't happen. If you think you've got something (i.e. you did some sort of systematic study), you're going down the wrong path. Any historical period you choose will be too vulnerable to oddball outcomes that have no bearing on any kind of legitimate risk premium assumption.*

The Capital Structure Weights

We're no coming down to the easy part. We need to establish the percentage weight for each capital item. Here are the rules:

- ShowVar(@Capital, DbtTot(0,qtr)+ PfdEquity(0,qtr) + ComEq(0,qtr))
    - *We define @Capital as the sum of the latest quarterly figures for DbtTot, PfdEquity and ComEq*
- ShowVar(@DbtWt,DbtTot(0,qtr)/@Capital)
- ShowVar(@PfdWt,PfdEquity(0,qtr)/@Capital)
- ShowVar(@EqWt,ComEq(0,qtr)/@Capital)
    - *This is a pretty basic trio. In each case Weight equals the capital item divided by @capital*

The Finish Line

- ShowVar(@CostCap,(@DbtWt*@DbtCost3)+(@PfdWt*@PfdCost)+(@EqWt*@CAPMRet))

    - *We're home! We're defining our final result, @CostCap as the sum of each capital item multiplied by its weight.*

That was one heck of a trip wasn't it.

Might you be tempted to ditch this and try a custom formula? You can't do it. There's a character limit and you can't use one custom formula as part of the computation of another. Even if we removed the character limit and let you go to town creating a full blown expression, you'd probably be so befuddled after setting up and de-bugging the function, we'd probably never see you again as you keep making futile efforts to log onto a site like portfolio629.gov or something like that.

It's not out of the question we might someday offer a WACC function. But it would have to one heck of a function with a lot of parameters: RF, RP, lower limit for cost of debt, upper limit for cost of debt, plugged in cost of debt when the database shows debt and no interest, tax rate, spread of preferred cost relative to debt cost, starting beta assumption, lower limit for beta, and upper limit for beta. And this is without considering that I used DbtTot instead of DbtLT which, financially speaking, would be a very valid choice. But companies report total interest, not LT interest, so you'd have to make some assumptions to derive an LT interest figure.

Seriously, ShowVar programming, once you get used to it, can really do wonders to enhance your use of portfolio123. And although this looks long, there's no obstacle to your continuing on in your screen or buy or sell rules with whatever other rules you want to use and plugging in @CostCap any time you have a formula that requires use of cost of capital (as we'll do later one as we move on to other topics).

And finally, recall that this is just the first practical solution I'm going to offer. It's not actually the one I use. It's a learning experience, to illustrate ShowVar programming and to convince you of the need for the approaches I'll offer later by showing the hoops you actually have to jump through in order to use what you might once have deemed a simple no-brainer formulation.

In Part 3, I'll offer a very simple and practical approach for WACC. It'll use much more spit and chewing gum than this model did. But it's shorter and I have used it in models that tested well. Then, in Part 4 (the final installment on WACC), we'll have some fun. We'll explore a completely outside the box approach inspired by some a very creative research from the academic community one of the professors being Stanford's Dr. Charles M.C. Lee, who developed a graduate level course (Alphanomics) around portfolio123, that being the focus of the textbook on which I'm currently working.

Appendix – WACC Using Basic CAPM for cost of Equity

Just to finish this section, here are the rules you'd use if you want a WACC computation based on conventionally-applied CAPM, with all its warts. If you wish, you can build a screen based on this approach and compare outcomes with the ones you get using my boundary-setting assumptions. (I won't include any annotations here. Having seen the approach offered before, you should recognize what this simpler set of rules is doing.)

- Showvar(@Int,ISNA(IntExpTTM*.65,0))

- Showvar(@Dbt,ISNA((DbtTot(0,qtr)+DbtTot(1,qtr)+DbtTot(2,qtr)+DbtTot(3,qtr)+DbtTot(4,qtr))/5,0))

- Showvar(@DbtCost,ISNA((@Int/@Dbt)*100,0))
- Showvar(@PfdCost,@DbtCost+1)
- Showvar(@CAPMRet,(close(0,#TNX)/10)+(Beta5Y*5))
- Showvar(@Capital, DbtTot(0,qtr)+PfdEquity(0,qtr)+ ComEq(0,qtr))
- Showvar(@DbtWt,DbtTot(0,qtr)/@Capital)
- Showvar(@PfdWt,PfdEquity(0,qtr)/@Capital)
- Showvar(@EqWt,ComEq(0,qtr)/@Capital)
- Showvar(@CostCap,(@DbtWt*@DbtCost)+(@PfdWt*@PfdCost)+(@EqWt*@CAPMRet))