Job Architecture - Structure Design

Nathan Wollack
https://www.nathanwollack.com/

Introduction

As someone who has spent their nearly 20+ year career in compensation, while having a passion for statistics and analysis, I have gone through the experience of working to update and create new salary structures. I authored this R script for others who may want a simple way to pull together market data and provide insights to assist in creating or updating a salary structure.

While Excel is a primary tool that compensation professionals use, it has limitations, one of which being ability to quickly do calculations with significant sized data sets. That is something that makes R ideal, as it is a powerful freeware application, that comes frills free.

Getting Started

To begin, you will want to install and load the desired packages. If you are new to R, it is an open source software that is popular with data miners, statisticians and data scientists. As it is open source, anyone who does development can create an extension to the base code, with functions and/or data sets. These extensions are called packages.

A package only needs to be installed once: however, simply installing does not mean it is available for use. A second step is needed to load the recently installed packages into your environment for use. The first set of code below is the chunk that installs the desired packages. If the package is already installed, R may generate an error message stating the packages are already loaded.

```
install.packages("dplyr", repos = "http://cran.us.r-project.org")
install.packages("readxl", repos = "http://cran.us.r-project.org")
install.packages("ggplot2", repos = "http://cran.us.r-project.org")
install.packages("stringr", repos = "http://cran.us.r-project.org")
install.packages("scales", repos = "http://cran.us.r-project.org")
```

This chunk of code will now load the packages just installed, for use in R.

```
library(openxlsx) #Used to read in Excel documents
library(dplyr) #Used to do data manipulation
library(ggplot2) #used for visualization design
library(stringr) #used for visualization design
library(scales) #used for visualization design
```

Installing the sample data set

For the purposes of this effort, I created a fake sample set, similar to what would be seen from a salary survey output. It is an excel document with 23 columns, with data points such as: Job Code, Job Function, Band, Level, Salary details, TTC details, and Performance Incentive details. All the data was randomly generated, with some logic included to mirror different pay details by level and by job function.

```
Survey<- read.xlsx("SampleSurvey.xlsx")
```

If the dataset has any blank values, it will cause an issue as you progress. There are really two options: remove from the start all rows that contain a blank value or address the blank values where applicable. My preference is the latter, because with salary survey data one row may have a value in field and a null value in another field.

Initial Structure output based upon survey data

One approach that is used for salary structure design is creating linear salary structures. What is a linear salary structure? That is where the midpoint progressions (percentage difference from a midpoint to the midpoint of the next highest grade) are constant throughout the whole structure.

First, we must set how many grades we want in this initial structure. For this example, I am using 15 grades.

```
NbrGrade<-15  #replace the number 15 with the number of grades you wish
              #to consider for the initial structure design
```

Now that we have set the number of grades, let's take the approach of looking at the data, and calculating the percentile data points for the lowest and highest grade. Another way to say it: if we wanted a five grade approach, when considering the full data set, what are the 10th and 90th percentile values of the Salary50th data point. Do those numbers seem extreme? The effort is to look for the midpoint of one fifth of the structure where we want grade 1 to be the Salary50th that captures from the lowest to the 20th percentile value, grade 2 to be the Salary50th from 20th to 40th percentile value and so on.

As we set the grades at 15, the below code will find 6.667th and 93.333th percentile value from the Salary50th data point. However, before running this chunk of code, note the inclusion of the na.rm = TRUE segment of the code. This is addressing the null / missing values.

```
TopBottom<-Survey%>%
  summarise(Lowest=quantile(Salary50th, probs = 1/(NbrGrade*2),na.rm = TRUE),
            Highest=quantile(Salary50th, probs = 1-(1/(NbrGrade*2)),na.rm = TRUE))
TopBottom
```

```
##     Lowest  Highest
## 1 29548.61 189400.3
```

As we see, the Lowest (6.667th) salary survey data point is 29,548.61 and the highest (93.33rd) salary survey data point is 189,400.30. The lowest value, 29,548.61 will be the midpoint for grade 1, and the highest value, 189,400.30 will be the midpoint for grade 15.

Now we need to get midpoints for grades 2-14. This is where we now have to find the percent that will get from 29,548.61 to 189,400.30 in 14 steps. This is done through the same way a compound interest number is obtained. The below chunk of code does just this; it looks at the midpoints we set for grade 1 and 15, and the number of grades desired and calculates out that Constant Midpoint Progression percentage.

```
ConstantMidPtProg <- as.numeric((TopBottom$Highest/TopBottom$Lowest)^(1/(NbrGrade-1))-1)
ConstantMidPtProg
```

```
## [1] 0.1419095
```

As you see, the percentage to get from the grade 1 to grade 15 in a constant manner is 14.19095%. If you are at all like me (a little bit of a skeptic), you are taking that percentage, going over to excel and typing in 32,192.12 and multiplying it by 1.134487 for 14 steps. Grade 15 from the calculation aligns perfectly. However, with mathematics, precision and rounding can always do a little bit of tripping items up, so there may be some instances where the calculation results in being a few cents off.

Now that we have the constant midpoint percentage, R can show you what the 15 grade structure would look like. The code will first create an item called Structure, which to start off with, will be just the value for grade 1 (which reminder, was the 6.67th percentile value of all the values in the Salary50th column). The code then builds the values for grades 2 through 15 as an item called Structure_Above; code then 'binds' the two datasets together and removes the Structure_Above data frame. It retitles the column header, adds a column that identifies the grade numbers, combines that with the midpoints, so all that remains related to the structures is a single data frame called structure, which has the midpoints for grades 1 through 15.

```
Midpoints<-TopBottom$Lowest
Midpoints_Above<-as.data.frame((1+ConstantMidPtProg)^(1:(NbrGrade-1))*TopBottom$Lowest)
Midpoints<-rbind(Midpoints,Midpoints_Above)
rm(Midpoints_Above)
colnames(Midpoints)[1]<-"Midpoint"
Grade<-as.factor(1*seq_len(nrow(Midpoints)))
cbind(Grade,Midpoints)
```

```
##    Grade  Midpoint
## 1      1  29548.61
## 2      2  33741.84
## 3      3  38530.13
## 4      4  43997.92
## 5      5  50241.64
## 6      6  57371.41
## 7      7  65512.95
## 8      8  74809.86
## 9      9  85426.09
## 10    10  97548.86
## 11    11 111391.97
## 12    12 127199.55
## 13    13 145250.37
## 14    14 165862.77
## 15    15 189400.27
```

So now you have your structure based upon a constant midpoint progression. However, companies do typically avoid having a salary structure that includes non-whole numbers and in many cases, like a smoothed approach to the numbers, such as round to the nearest multiple of x. We have a code for that.

Smoothing the Initial Structure

This is a pretty easy process to do. We simply have to create a function, that rounds to the nearest multiple, set the desired multiple, and get the output of the rounded structure.

```
mround <- function(number, multiple) multiple * round(number/multiple)
Midpoints_Rounded<-mround(Midpoints,100)
cbind(Grade,Midpoints_Rounded)
```

```
##    Grade Midpoint
## 1      1    29500
## 2      2    33700
## 3      3    38500
## 4      4    44000
## 5      5    50200
## 6      6    57400
## 7      7    65500
## 8      8    74800
## 9      9    85400
## 10    10    97500
## 11    11   111400
## 12    12   127200
## 13    13   145300
## 14    14   165900
## 15    15   189400
```

Setting Minimum and Maximum

So if the midpoints are set, it is now time to set the minimum and the maximum.

A common misstep I have seen is where the variance from minimum to midpoint does not equal the variance from midpoint to maximum.

Why does this occur? Because of the rounding to the nearest multiple on both sides. I shared above that rounding can cause issues and that is the case here.

That is why, I am an advocate for setting the minimum and then taking the variance from minimum to midpoint, and adding that variance to the midpoint to get the maximum. This will result in the midpoint being exactly 50% when doing position in range calculation.

Using the below code, a single range width can be set. There are differing views on this: as some companies prefer a set range width regardless of level, while others have an approach where the range width expands the higher up the grade. I am on team expanding range-width.

```
RangeWidth<-.4
```

Now that the range width has been set, this code chunk will build the minimum and maximum, as well as show you what the compa-ratio is for those values.

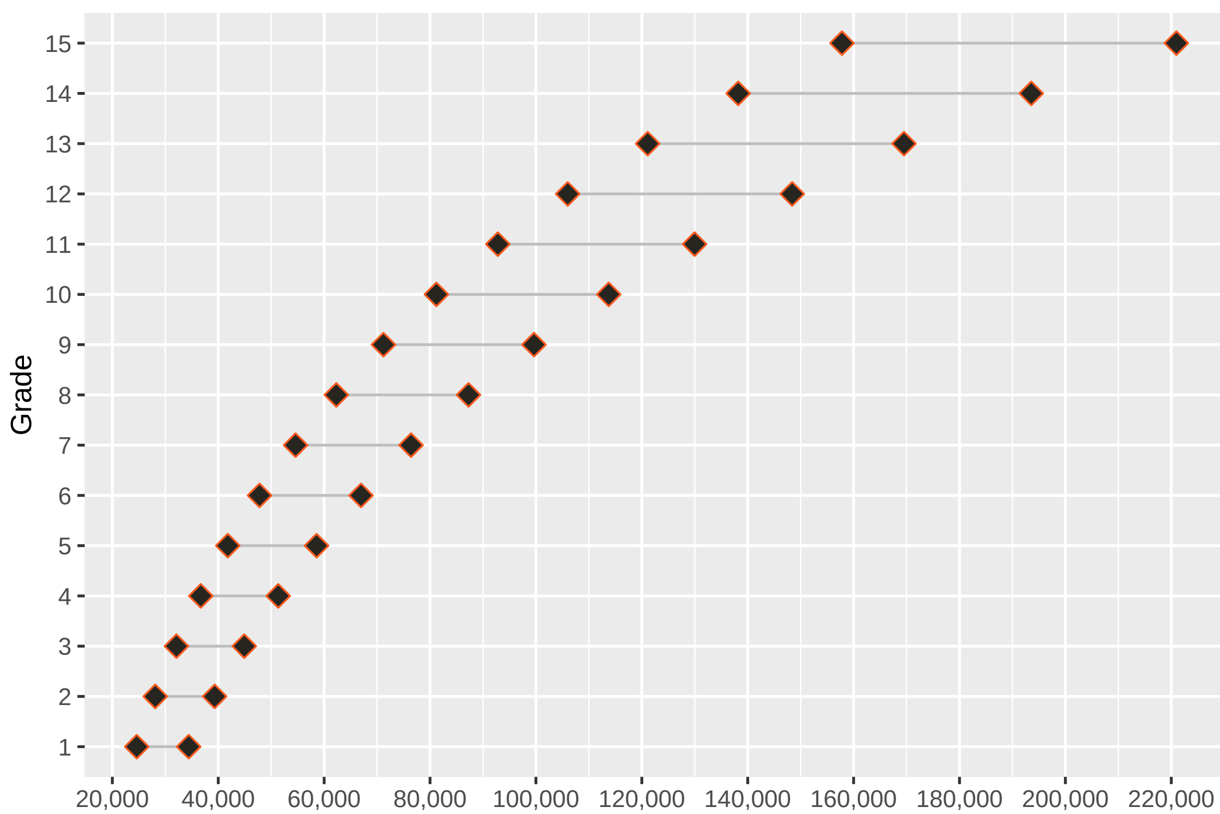
```
Structure_Rounded<-Midpoints_Rounded%>%mutate(Minimum=(Midpoint/(1+(.5*RangeWidth))))
attach(Structure_Rounded)
Structure_Rounded<-mround(Structure_Rounded,100)
Structure_Rounded$Maximum<-Midpoint-Minimum+Midpoint
Structure_Rounded<-Structure_Rounded[,c(2,1,3)]
Structure_Rounded<-cbind(Grade,Structure_Rounded)
Structure_Rounded$MinCR<-round(Minimum/Midpoint,4)
Structure_Rounded$MaxCR<-round(Structure_Rounded$Maximum/Midpoint,4)
Structure_Rounded
```

```
##    Grade Minimum Midpoint  Maximum  MinCR  MaxCR
## 1      1   24600    29500 34416.67 0.8333 1.1667
## 2      2   28100    33700 39316.67 0.8333 1.1667
## 3      3   32100    38500 44916.67 0.8333 1.1667
```

```
## 4      4   36700    44000  51333.33 0.8333 1.1667
## 5      5   41800    50200  58566.67 0.8333 1.1667
## 6      6   47800    57400  66966.67 0.8333 1.1667
## 7      7   54600    65500  76416.67 0.8333 1.1667
## 8      8   62300    74800  87266.67 0.8333 1.1667
## 9      9   71200    85400  99633.33 0.8333 1.1667
## 10    10   81200    97500 113750.00 0.8333 1.1667
## 11    11   92800   111400 129966.67 0.8333 1.1667
## 12    12  106000   127200 148400.00 0.8333 1.1667
## 13    13  121100   145300 169516.67 0.8333 1.1667
## 14    14  138200   165900 193550.00 0.8333 1.1667
## 15    15  157800   189400 220966.67 0.8333 1.1667
```

The 15 grade range now has minimums and maximums, based upon a consistent range width. For some, this is enough. For others, they may want to see how these ranges look visually. I have constructed a lollipop chart, that shows how each range aligns to one another visually.

```
ggplot(Structure_Rounded) +
  geom_segment( aes(x=Grade, xend=Grade, y=Minimum, yend=Maximum), color="grey") +
  geom_point( aes(x=Grade, y=Minimum ), color="#FD5A1E", size=3,shape=23,fill="#27251F") +
  geom_point( aes(x=Grade, y=Maximum ), color="#FD5A1E", size=3,shape=23,fill="#27251F") +
  coord_flip()+
  ylab("")+
  xlab("Grade")+
  scale_y_continuous(label=comma, breaks=seq(0,350000,20000))
```



As was discussed above, there is a prevalent thought that as the grades get higher, the range width should expand. A common reasoning for this is that as someone starts lower in the organization, their opportunity to advance throughout the range is improved. So the higher up an employee gets in the grading structure, longevity in the role becomes ideal and as a response, progression upward in the salary grade structure is slowed. While the thought of widening range widths as progression through the structure occurs, there is differing opinions on just what those range widths should be. A good blog related to this is from People-Centre, https://peoplecentre.wordpress.com/2014/09/29/salary-structure-range-spread/ where it is is clear that there is nothing consistent around perspectives on this.

Thus, I am setting up 3 range widths: one at 40% for lower graded roles (grades 1-5); one at 55% for middle graded roles (grades 5-10); one at 70% for higher graded roles (11-15).

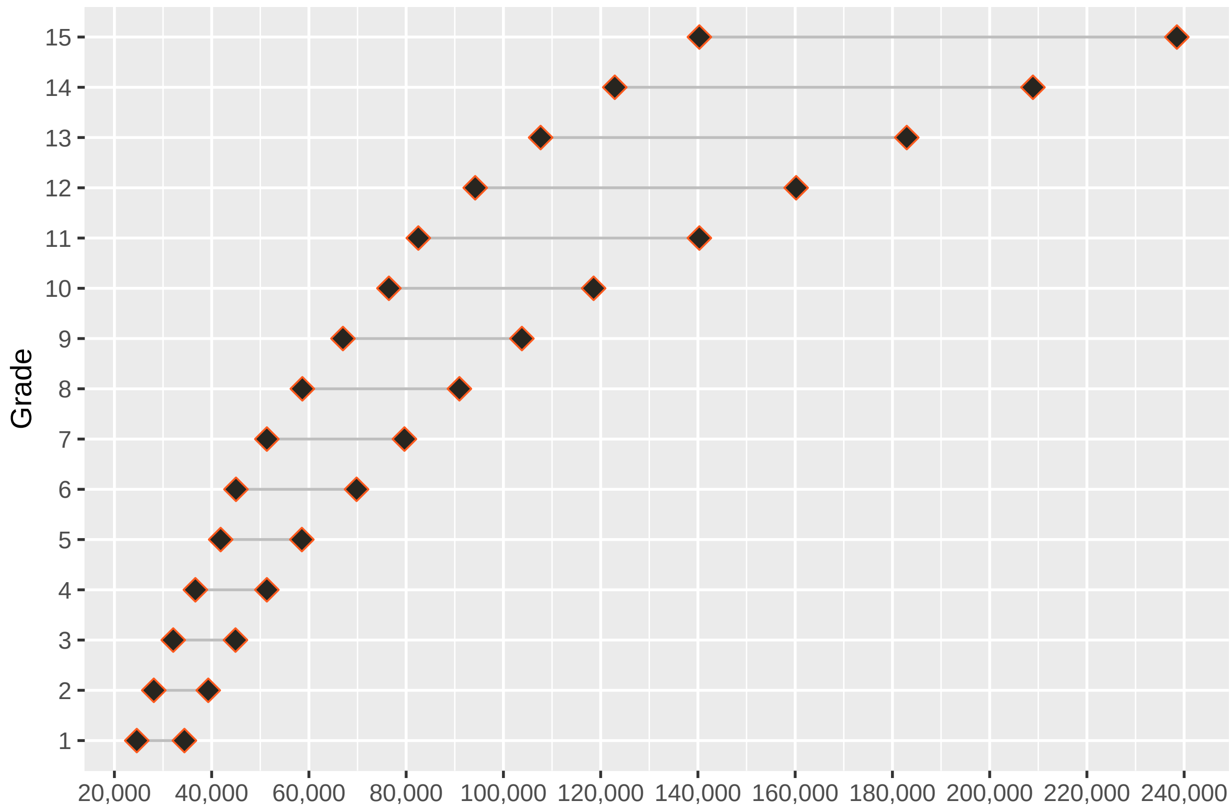
```
RangeWidth1<-.4
RangeWidth2<-.55
RangeWidth3<-.70

Structure_Rounded_Var_RW<-Structure_Rounded[,c(2,3,4)]
Structure_Rounded_Var_RW = Structure_Rounded_Var_RW%>%
  mutate(Minimum = ifelse(as.integer(Grade)<6, Midpoint/(1+(.5*RangeWidth1)),
                   ifelse(as.integer(Grade)<11,Midpoint/
                   (1+(.5*RangeWidth2)),Midpoint/(1+(.5*RangeWidth3)))))
Structure_Rounded_Var_RW<-mround(Structure_Rounded_Var_RW,50)
Structure_Rounded_Var_RW$Maximum<-Structure_Rounded_Var_RW$Midpoint-
  Structure_Rounded_Var_RW$Minimum+Structure_Rounded_Var_RW$Midpoint
Structure_Rounded_Var_RW<-cbind(Grade,Structure_Rounded_Var_RW)
Structure_Rounded_Var_RW$MinCR<-round(Structure_Rounded_Var_RW$Minimum/
                                        Structure_Rounded_Var_RW$Midpoint,4)
Structure_Rounded_Var_RW$MaxCR<-round(Structure_Rounded_Var_RW$Maximum/
                                        Structure_Rounded_Var_RW$Midpoint,4)
Structure_Rounded_Var_RW
```

```
##    Grade Minimum Midpoint Maximum  MinCR  MaxCR
## 1      1   24600    29500   34400 0.8339 1.1661
## 2      2   28100    33700   39300 0.8338 1.1662
## 3      3   32100    38500   44900 0.8338 1.1662
## 4      4   36650    44000   51350 0.8330 1.1670
## 5      5   41850    50200   58550 0.8337 1.1663
## 6      6   45000    57400   69800 0.7840 1.2160
## 7      7   51350    65500   79650 0.7840 1.2160
## 8      8   58650    74800   90950 0.7841 1.2159
## 9      9   67000    85400  103800 0.7845 1.2155
## 10    10   76450    97500  118550 0.7841 1.2159
## 11    11   82500   111400  140300 0.7406 1.2594
## 12    12   94200   127200  160200 0.7406 1.2594
## 13    13  107650   145300  182950 0.7409 1.2591
## 14    14  122900   165900  208900 0.7408 1.2592
## 15    15  140300   189400  238500 0.7408 1.2592
```

Similar to the other structure, let's look at a visualization.

```
ggplot(Structure_Rounded_Var_RW) +
  geom_segment(aes(x=Grade, xend=Grade, y=Minimum, yend=Maximum), color="grey") +
  geom_point(aes(x=Grade, y=Minimum ),color="#FD5A1E",size=3,shape=23,fill="#27251F") +
  geom_point(aes(x=Grade, y=Maximum ),color="#FD5A1E",size=3,shape=23,fill="#27251F") +
```

```
  coord_flip()+
  ylab("")+
  xlab("Grade")+
  scale_y_continuous(label=comma, breaks=seq(0,350000,20000))
```



Something may initially look a little odd about this image, as the minimum line marker seem a bit out of place at certain levels. Those places coincide with where the range width was adjusted (grades 6 and 11).

Setting Structures based upon alignment of Band & Level to Grade

A company may decide that setting a linear structure does not align to their philosophy and may pursue an approach of aligning their grades to survey market matches, based upon the survey's band (support, professional, management) and the level within those bands (i.e. Management 1, Professional 2). This can also be done through R code. The first thing to consider is how well the band / levels align and that can help determine number of grades in the structure. This is very much the art side of compensation.

Looking at the aggregation of the data can be done easily by running the below chunk, which takes the survey data and does an simple average of the Salary50th data. In excel terms, it is just like the AverageIf function.

```
SalaryByBand_Level<-Survey%>%
  group_by(Band_Level)%>%
  summarise(Salary=mean(Salary50th,na.rm = TRUE))%>%
  arrange(Salary)

SalaryByBand_Level
```

```
## # A tibble: 25 x 2
##    Band_Level Salary
##    <chr>       <dbl>
##  1 O1         29059.
##  2 O2         32927.
##  3 A1         37354.
##  4 O3         37755.
##  5 A2         43193.
##  6 O4         43360.
##  7 T1         49461.
##  8 A3         49849.
##  9 O5         50672.
## 10 T2         56958.
## # ... with 15 more rows
```

As was noted prior, this is where some art comes into play. I might take out a scratch piece of paper and start to build an alignment document of grouped band_level terms. In looking at the first two rows of data, it may seem like considering grouping the O1 and O2 based upon this data. However, a quick mental calculation tells me that it differs by greater than 10%. Additionally, it might be helpful to add to this summary the average count for org and incumbent for the survey roles in the band_level. With a few additional lines in the chunk, we can achieve this.

```
SalaryByBand_Level<-Survey%>%
  group_by(Band_Level)%>%
  summarise(Salary=mean(Salary50th,na.rm = TRUE),
            OrgCount=mean(SalaryOrgCount,na.rm = TRUE),
            IncCount=mean(SalaryIncCount,na.rm = TRUE))%>%
  arrange(Salary)%>%
  mutate(Progression = (Salary / lag(Salary))-1)

SalaryByBand_Level
```

```
## # A tibble: 25 x 5
##    Band_Level Salary OrgCount IncCount Progression
##    <chr>       <dbl>    <dbl>    <dbl>       <dbl>
##  1 O1         29059.     33.9     497.    NA
##  2 O2         32927.     34.7     307.     0.133
##  3 A1         37354.     35.7     524.     0.134
##  4 O3         37755.     36.2     410.     0.0107
##  5 A2         43193.     36.7     329.     0.144
##  6 O4         43360.     30.4     363.     0.00388
##  7 T1         49461.     30.7     285.     0.141
##  8 A3         49849.     38.9     435.     0.00784
##  9 O5         50672.     22.8     275.     0.0165
## 10 T2         56958.     30.0     379.     0.124
## # ... with 15 more rows
```

As O1 is the lowest, there is no progression calculation, however, we see that O2 is ~13.3 higher than O1. We also see a ~13.4 jump from O2 to the next highest aggregate, A1. So on that alignment scratch pad, O1 = grade 1, O2 = grade 2 and A1 = grade 3.

We do see that O3 is only ~.011 higher than the A1, so O3 best aligns to grade 3. You continue to do this practice until you feel fairly comfortable and all Band_Levels are aligned to a grade.

When looking at this data, an approach could be to do 12 grades. Now the coding needs to occur to put the grades into the data set. The first part of the below chunk looks for all unique values of Band_Level (25 unique Band_Levels).

```
BandLevel_Unique<-unique(Survey$Band_Level)
BandLevel_Unique
```

```
##  [1] "A1" "A2" "A3" "A4" "A5" "L1" "L2" "L3" "L4" "L5" "O1" "O2" "O3" "O4" "O5"
## [16] "P1" "P2" "P3" "P4" "P5" "T1" "T2" "T3" "T4" "T5"
```

The next part is to align a numeric grade to these 25 Band_Levels. The code will create a vector called Grade_UniqueLevel and begin to assign, in parallel order to the above BandLevel_Unique vector, the grades. First band_level is A1, which aligned to grade 3; next is A2 and align to grade 4 and continue this process until you have aligned all 25 band_level to a grade.

```
Grade_UniqueLevel<-c(3,4,5,6,7,8,9,10,11,12,1,2,3,4,5,6,7,8,9,10,5,6,7,8,9)
Grade_UniqueLevel
```

```
##  [1]  3  4  5  6  7  8  9 10 11 12  1  2  3  4  5  6  7  8  9 10  5  6  7  8  9
```

We now combine the two vectors into a dataframe, called Grade_Band_Level, do some cleanup on column titles and see the results.

```
Grade_BandLevel<-data.frame(Grade_UniqueLevel,BandLevel_Unique)
Grade_BandLevel$Grade_UniqueLevel<-as.factor(Grade_BandLevel$Grade_UniqueLevel)
colnames(Grade_BandLevel)<-c("Grade","Band_Level")
Grade_BandLevel
```

```
##    Grade Band_Level
## 1      3         A1
## 2      4         A2
## 3      5         A3
## 4      6         A4
## 5      7         A5
## 6      8         L1
## 7      9         L2
## 8     10         L3
## 9     11         L4
## 10    12         L5
## 11     1         O1
## 12     2         O2
## 13     3         O3
## 14     4         O4
## 15     5         O5
## 16     6         P1
## 17     7         P2
## 18     8         P3
## 19     9         P4
## 20    10         P5
```

```
## 21      5          T1
## 22      6          T2
## 23      7          T3
## 24      8          T4
## 25      9          T5
```

Now that we have the alignment of grade to band_level, we need to bring the alignment back to the Survey data. A simple merge code achieves this.

```
Survey<-merge(Survey,Grade_BandLevel,by="Band_Level")
```

We have our alignment complete, and it is now aligned to the survey data. The data can now be pulled into a summary.

However, we can look at the alignments in a few different ways:

- Take the simple average of the Salary50th, where a role that has limited incumbents or organizations aligned to it brings as much impact to the structure design as a role that is quite prevalent or
- Do weighted averages for both incumbents and organization

Why not have all three?

```
Market_Midpoints<-Survey%>%
  group_by(Grade)%>%
  summarise(SimpleAvgOf50th=mean(Salary50th,na.rm = TRUE),
            WeightedAvgof50_Inc=weighted.mean(Salary50th, SalaryIncCount,na.rm = TRUE),
            WeightedAvgof50_Org=weighted.mean(Salary50th, SalaryOrgCount,na.rm = TRUE))%>%
            mutate(ProgSimpleAvg=(SimpleAvgOf50th/lag(SimpleAvgOf50th))-1,
               ProgWeightedInc=(WeightedAvgof50_Inc/lag(WeightedAvgof50_Inc))-1,
               ProgWeightedOrg=(WeightedAvgof50_Org/lag(WeightedAvgof50_Org))-1)

Market_Midpoints
```

```
## # A tibble: 12 x 7
##    Grade SimpleAvgOf50th WeightedAvgof50_Inc WeightedAvgof50_Org ProgSimpleAvg
##    <fct>           <dbl>               <dbl>               <dbl>         <dbl>
##  1 1              29059.              29169.              29166.        NA
##  2 2              32927.              33043.              33049.         0.133
##  3 3              37554.              37680.              37682.         0.141
##  4 4              43276.              43530.              43470.         0.152
##  5 5              49988.              50126.              50128.         0.155
##  6 6              58301.              58503.              58512.         0.166
##  7 7              70153.              70872.              70801.         0.203
##  8 8              88863.              88959.              89083.         0.267
##  9 9             111401.             111708.             111768.         0.254
## 10 10            141258.             140877.             140958.         0.268
## 11 11            167729.             168554.             168565.         0.187
## 12 12            207821.             208936.             208802.         0.239
## # ... with 2 more variables: ProgWeightedInc <dbl>, ProgWeightedOrg <dbl>
```

There is now market based salary midpoints, with looks by simple average and weighted by incumbents and organizations, along progressions by the three midpoint views. Similar to the linear midpoint progression view, we can initially smooth using the same logic as above.

```r
Market_Midpoints_Rounded<-mround(Market_Midpoints[,c(2:4)],100)
Grade_Market<-as.factor(1:12)
Market_Midpoints_Rounded<-cbind(Grade_Market,Market_Midpoints_Rounded)
colnames(Market_Midpoints_Rounded)[1]<-"Grade"
Market_Midpoints_Rounded$Grade<-Market_Midpoints_Rounded$Grade
Market_Midpoints_Rounded
```

```
##    Grade SimpleAvgOf50th WeightedAvgof50_Inc WeightedAvgof50_Org
## 1      1           29100               29200               29200
## 2      2           32900               33000               33000
## 3      3           37600               37700               37700
## 4      4           43300               43500               43500
## 5      5           50000               50100               50100
## 6      6           58300               58500               58500
## 7      7           70200               70900               70800
## 8      8           88900               89000               89100
## 9      9          111400              111700              111800
## 10    10          141300              140900              141000
## 11    11          167700              168600              168600
## 12    12          207800              208900              208800
```

With the midpoints shown, one can select which midpoints they wish to build a structure based upon. My preference is to use incumbent weighted, which is fairly similar to the others; however, one I believe provides a better assessment of the market.

The below chunk takes the midpoint for incumbent weighted and builds a structure based a constant range width (same range width used above or 40%)

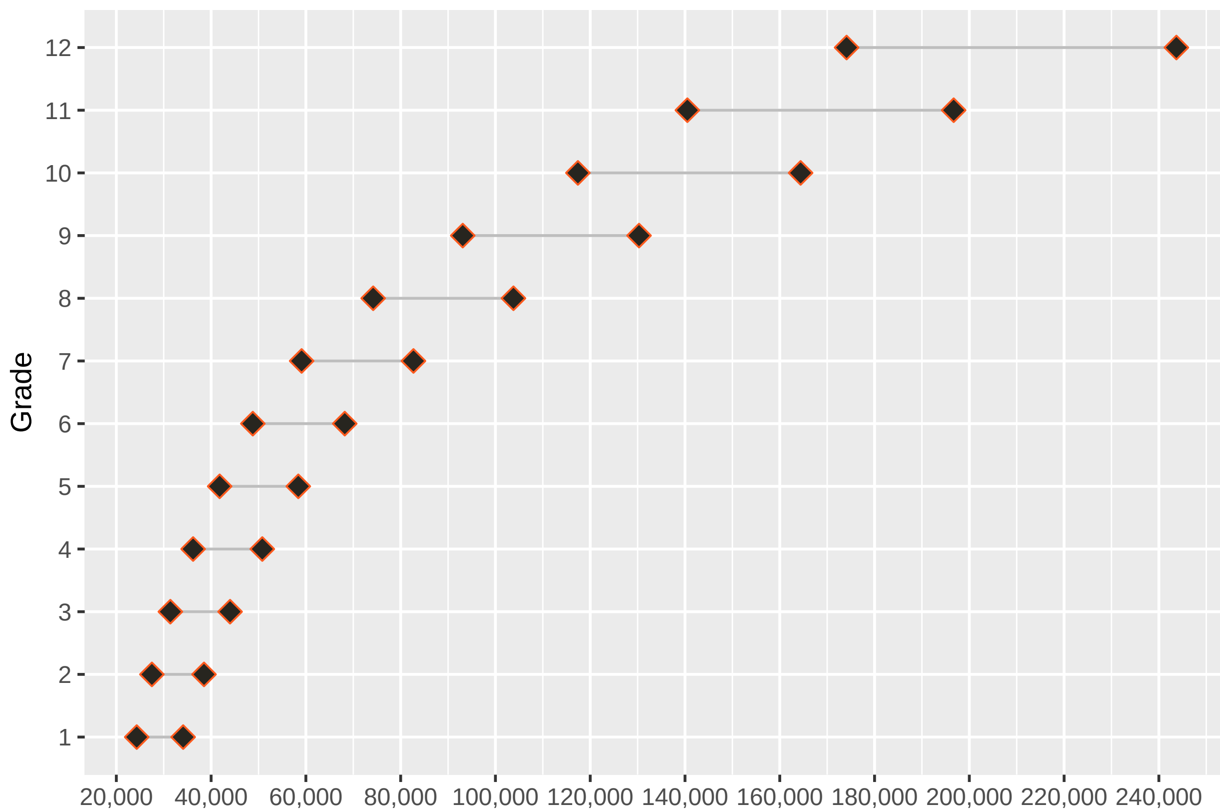
```r
Market_Midpoints_Rounded_Inc<-as.data.frame(Market_Midpoints_Rounded[,c(3)])
colnames(Market_Midpoints_Rounded_Inc)[1]<-"Midpoint"
Market_Structure_Rounded<-Market_Midpoints_Rounded_Inc%>%mutate(Minimum=(Midpoint/
                                                         (1+(.5*RangeWidth))))
Market_Structure_Rounded<-mround(Market_Structure_Rounded,100)
Market_Structure_Rounded$Maximum<-Market_Structure_Rounded$Midpoint-
  Market_Structure_Rounded$Minimum+Market_Structure_Rounded$Midpoint

Market_Structure_Rounded<-Market_Structure_Rounded[,c(2,1,3)]
Market_Structure_Rounded<-cbind(Grade_Market,Market_Structure_Rounded)
colnames(Market_Structure_Rounded)[1]<-"Grade"
Market_Structure_Rounded$MinCR<-round(Market_Structure_Rounded$Minimum/
                                        Market_Structure_Rounded$Midpoint,4)
Market_Structure_Rounded$MaxCR<-round(Market_Structure_Rounded$Maximum/
                                        Market_Structure_Rounded$Midpoint,4)
Market_Structure_Rounded
```

```
##    Grade Minimum Midpoint Maximum  MinCR  MaxCR
## 1      1   24300    29200   34100 0.8322 1.1678
## 2      2   27500    33000   38500 0.8333 1.1667
## 3      3   31400    37700   44000 0.8329 1.1671
## 4      4   36200    43500   50800 0.8322 1.1678
## 5      5   41800    50100   58400 0.8343 1.1657
## 6      6   48800    58500   68200 0.8342 1.1658
## 7      7   59100    70900   82700 0.8336 1.1664
```

```
## 8      8   74200    89000   103800 0.8337 1.1663
## 9      9   93100   111700   130300 0.8335 1.1665
## 10    10  117400   140900   164400 0.8332 1.1668
## 11    11  140500   168600   196700 0.8333 1.1667
## 12    12  174100   208900   243700 0.8334 1.1666
```

We can show a visualization of the range, such as was done earlier in this document.

```
ggplot(Market_Structure_Rounded) +
  geom_segment(aes(x=Grade, xend=Grade, y=Minimum, yend=Maximum), color="grey") +
  geom_point(aes(x=Grade, y=Minimum ), color="#FD5A1E",size=3,shape=23,fill="#27251F") +
  geom_point(aes(x=Grade, y=Maximum ), color="#FD5A1E",size=3,shape=23,fill="#27251F") +
  coord_flip()+
  ylab("")+
  xlab("Grade")+
  scale_y_continuous(label=comma, breaks=seq(0,350000,20000))
```



The constant range width again is one approach, however, as noted above, many companies use a varied range width for different grades. The widths are adjusted for this market look, at 45% for grades 1-4, 60% for grade 5-8 and 75% for grades 9-12.

```
MktRng1<-.45
MktRng2<-.60
MktRng3<-.75

MktStruc_Rounded_Var_RW<-Market_Structure_Rounded[,c(2,3,4)]
```

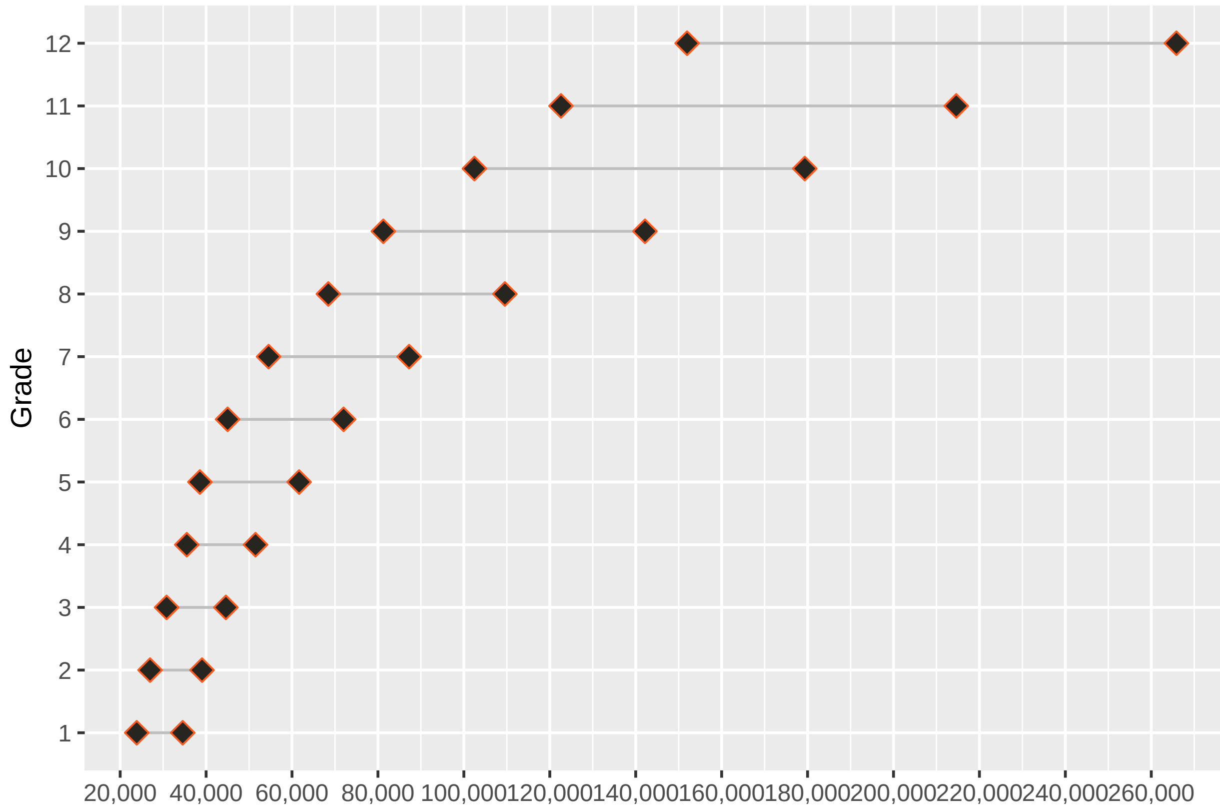
```r
MktStruc_Rounded_Var_RW = MktStruc_Rounded_Var_RW%>%
  mutate(Minimum = ifelse(as.integer(Grade_Market)<5, Midpoint/(1+(.5*MktRng1)),
                     ifelse(as.integer(Grade_Market)<9,Midpoint/
                     (1+(.5*MktRng2)),Midpoint/(1+(.5*MktRng3)))))
MktStruc_Rounded_Var_RW<-mround(MktStruc_Rounded_Var_RW,50)
MktStruc_Rounded_Var_RW$Maximum<-MktStruc_Rounded_Var_RW$Midpoint-
  MktStruc_Rounded_Var_RW$Minimum+MktStruc_Rounded_Var_RW$Midpoint
MktStruc_Rounded_Var_RW<-cbind(Grade_Market,MktStruc_Rounded_Var_RW)
colnames(MktStruc_Rounded_Var_RW)[1]<-"Grade"
MktStruc_Rounded_Var_RW$MinCR<-round(MktStruc_Rounded_Var_RW$Minimum/
                                         MktStruc_Rounded_Var_RW$Midpoint,4)
MktStruc_Rounded_Var_RW$MaxCR<-round(MktStruc_Rounded_Var_RW$Maximum/
                                         MktStruc_Rounded_Var_RW$Midpoint,4)
MktStruc_Rounded_Var_RW
```

```
##    Grade Minimum Midpoint Maximum  MinCR  MaxCR
## 1      1   23850    29200   34550 0.8168 1.1832
## 2      2   26950    33000   39050 0.8167 1.1833
## 3      3   30800    37700   44600 0.8170 1.1830
## 4      4   35500    43500   51500 0.8161 1.1839
## 5      5   38550    50100   61650 0.7695 1.2305
## 6      6   45000    58500   72000 0.7692 1.2308
## 7      7   54550    70900   87250 0.7694 1.2306
## 8      8   68450    89000  109550 0.7691 1.2309
## 9      9   81250   111700  142150 0.7274 1.2726
## 10    10  102450   140900  179350 0.7271 1.2729
## 11    11  122600   168600  214600 0.7272 1.2728
## 12    12  151950   208900  265850 0.7274 1.2726
```

Considering the new range widths, we can have a look at the structures visually. Similar to the conversation on range widths is the idea of range overlap.

One thought is that the overlap should be minimal, with perhaps 2 grades above having overlap. As you see below, at the lower levels, this proposed structure looks to have 2 or maybe 3 levels with overlap. In the above table, you can see that with the exception of grade 2 and grade 5, all other grades do stay at the two grade overlap. However, if an approach is taken to widen the ranges from the 45%, 60%, 75%, there will be more levels that overlap. Therefore, it is a balance of having ranges that are not too restrictive, while not having them too wide which can cause concerns, such as trying to manage pay equity.

```r
ggplot(MktStruc_Rounded_Var_RW) +
  geom_segment(aes(x=Grade, xend=Grade, y=Minimum, yend=Maximum), color="grey")+
  geom_point(aes(x=Grade, y=Minimum ),color="#FD5A1E",size=3,shape=23,fill="#27251F")+
  geom_point(aes(x=Grade, y=Maximum ),color="#FD5A1E",size=3,shape=23,fill="#27251F")+
  coord_flip()+
  ylab("")+
  xlab("Grade")+
  scale_y_continuous(label=comma, breaks=seq(0,350000,20000))
```



Differing structure by function - Band and Level approach

An approach that does get used in some salary structure design is functional or premium structures. Reality is the median salary for a professional level 3 role can differ greatly depending on the type of role; some functions are paid higher than others. So taking a one size fits all approach may not be supported and a request may come in to consider a few different structures that are built to group similar functions related to salary. There is an ability to generate structures through an R script to provide the different structures. However, identifying those functions to be segmented from the 'baseline' structure is another instance where it becomes an art versus a science.

One way to consider looking at this is to compare on a level role by role basis, how does the salary50th value compare to the average Salary50th by the Band_level combination. The below code provides this information and by using the 'head' function and selecting only a few of the relevant columns, we can see in general how this approach looks.

```
PctOfMedian<-Survey%>%
  group_by(Band_Level)%>%
  mutate(MedianSalary_BandLevel=mean(Salary50th,na.rm = TRUE))%>%
  ungroup()%>%
  group_by(Job_Function,Band_Level)%>%
  mutate(PctMedian_Func=mean(Salary50th,na.rm = TRUE)/MedianSalary_BandLevel)

head(PctOfMedian[,c(2,10,25:26)])
```

```
## # A tibble: 6 x 4
```

```
##   JobCode Salary50th MedianSalary_BandLevel PctMedian_Func
##   <chr>        <dbl>                  <dbl>          <dbl>
## 1 AAA-A1      34558.                 37354.          0.925
## 2 ABI-A1      35009.                 37354.          0.937
## 3 AGO-A1         NA                  37354.        NaN
## 4 AGT-A1      35073.                 37354.          0.939
## 5 ACV-A1      42010.                 37354.          1.12
## 6 ABN-A1      38406.                 37354.          1.03
```

This table shows that for Function AAA and band/level A1, the Salary50th is 34558.39 and the Median Salary for the A1 band level is 37353.76. Therefore, the AAA-A1 job is ~92.5% of the median base salary for an A1 band and level.

We can take this data and the aggregation of all functions and band / level combinations to get a sense of how that function compares.

```
PctOfMedian_Summary<-PctOfMedian%>%
  group_by(Job_Function)%>%
  summarise(PctMedian_Func=mean(Salary50th/MedianSalary_BandLevel, na.rm=TRUE))%>%
  arrange(PctMedian_Func)

head(PctOfMedian_Summary)
```

```
## # A tibble: 6 x 2
##   Job_Function PctMedian_Func
##   <chr>                 <dbl>
## 1 AHQ                   0.823
## 2 AGI                   0.826
## 3 ADS                   0.832
## 4 AEG                   0.833
## 5 ADT                   0.833
## 6 AHI                   0.835
```

We see that for function AHQ, in aggregation, it is 82.26% of the median Salary50th. A decision needs to be mad around how many structures are desired; Should it be 3 structures with a reduced, baseline and premium? Should it be 2 with baseline and premium? For sake of this exercise, lets consider 3 and take the approach of all functions <90% are reduced, 90%-110% are baseline and >110% are premium.

First, we simply need to bring in the PctMedian_Func column into "Survey", our base data frame.

```
Survey<-merge(Survey,PctOfMedian_Summary,by="Job_Function")
```

Next, we can add a column that uses an If then statement to put in the structure name.

```
Survey$Structure<-ifelse(Survey$PctMedian_Func<.90,"Reduced",
                         ifelse(Survey$PctMedian_Func<1.10,"Baseline","Premium"))
```

So structures have been assigned, by slightly modifying a code used above (by adding Structure to the Group_By line), the new structures can be seen.

```
Market_Midpoints_Structures<-Survey%>%
  group_by(Structure,Grade)%>%
  filter(!is.na(Structure))%>%
```

```
  summarise(SimpleAvgOf50th=mean(Salary50th,na.rm = TRUE),
            WeightedAvgof50_Inc=weighted.mean(Salary50th, SalaryIncCount,na.rm = TRUE),
            WeightedAvgof50_Org=weighted.mean(Salary50th, SalaryOrgCount,na.rm = TRUE))%>%
            mutate(ProgSimpleAvg = (SimpleAvgOf50th / lag(SimpleAvgOf50th))-1,
                ProgWeightedInc = (WeightedAvgof50_Inc / lag(WeightedAvgof50_Inc))-1,
                ProgWeightedOrg = (WeightedAvgof50_Org / lag(WeightedAvgof50_Org))-1)
```

```
## `summarise()` has grouped output by 'Structure'. You can override using the
## `.groups` argument.
```

```
Market_Midpoints_Structures
```

```
## # A tibble: 36 x 8
## # Groups:   Structure [3]
##    Structure Grade SimpleAvgOf50th WeightedAvgof50_Inc WeightedAvgof50_Org
##    <chr>     <fct>           <dbl>               <dbl>               <dbl>
##  1 Baseline  1              29035.              29092.              29076.
##  2 Baseline  2              32811.              32792.              32771.
##  3 Baseline  3              37436.              37460.              37449.
##  4 Baseline  4              43078.              43079.              43049.
##  5 Baseline  5              49673.              49759.              49736.
##  6 Baseline  6              57829.              57798.              57882.
##  7 Baseline  7              69559.              70106.              70070.
##  8 Baseline  8              88232.              88148.              88247.
##  9 Baseline  9             110889.             110994.             111018.
## 10 Baseline  10            140981.             140137.             140288.
## # ... with 26 more rows, and 3 more variables: ProgSimpleAvg <dbl>,
## #   ProgWeightedInc <dbl>, ProgWeightedOrg <dbl>
```

By doing the same exercise to these structures that was done above for the overall structure, can see smoothed midpoints and the different minimums and maximums. However, it is best to separate the three structures out as their own data frames.

```
Market_Midpoints_Reduced<-subset(Market_Midpoints_Structures,Structure=="Reduced")
Market_Midpoints_Baseline<-subset(Market_Midpoints_Structures,Structure=="Baseline")
Market_Midpoints_Premium<-subset(Market_Midpoints_Structures,Structure=="Premium")
```

First, lets round the reduced structure.

```
Market_Midpoints_Reduc_Rnd<-mround(Market_Midpoints_Reduced[,c(3:5)],100)
Grade_Market<-as.factor(1:12)
Market_Midpoints_Reduc_Rnd<-cbind(Grade_Market,Market_Midpoints_Reduc_Rnd)
colnames(Market_Midpoints_Reduc_Rnd)[1]<-"Grade"
Market_Midpoints_Reduc_Rnd
```

```
##    Grade SimpleAvgOf50th WeightedAvgof50_Inc WeightedAvgof50_Org
## 1      1           25300               25300               25300
## 2      2           28400               28500               28500
## 3      3           32400               32300               32300
## 4      4           37300               37500               37400
## 5      5           43400               43300               43300
```

```
## 6       6           50700             50700             50800
## 7       7           61000             61500             61500
## 8       8           76800             76500             76600
## 9       9           96300             96200             96100
## 10     10          122200            121000            121100
## 11     11          144700            144700            144700
## 12     12          177400            177400            177100
```

Next is the baseline structure.

```
Market_Midpoints_BLine_Rnd<-mround(Market_Midpoints_Baseline[,c(3:5)],100)
Grade_Market<-as.factor(1:12)
Market_Midpoints_BLine_Rnd<-cbind(Grade_Market,Market_Midpoints_BLine_Rnd)
colnames(Market_Midpoints_BLine_Rnd)[1]<-"Grade"
Market_Midpoints_BLine_Rnd
```

```
##    Grade SimpleAvgOf50th WeightedAvgof50_Inc WeightedAvgof50_Org
## 1      1           29000               29100               29100
## 2      2           32800               32800               32800
## 3      3           37400               37500               37400
## 4      4           43100               43100               43000
## 5      5           49700               49800               49700
## 6      6           57800               57800               57900
## 7      7           69600               70100               70100
## 8      8           88200               88100               88200
## 9      9          110900              111000              111000
## 10    10          141000              140100              140300
## 11    11          166500              166600              166800
## 12    12          207900              208800              208600
```

Last is the premium structure.

```
Market_Midpoints_Prem_Rnd<-mround(Market_Midpoints_Premium[,c(3:5)],100)
Grade_Market<-as.factor(1:12)
Market_Midpoints_Prem_Rnd<-cbind(Grade_Market,Market_Midpoints_Prem_Rnd)
colnames(Market_Midpoints_Prem_Rnd)[1]<-"Grade"
Market_Midpoints_Prem_Rnd
```

```
##    Grade SimpleAvgOf50th WeightedAvgof50_Inc WeightedAvgof50_Org
## 1      1           33100               33100               33100
## 2      2           37800               38000               38000
## 3      3           43100               43300               43300
## 4      4           50000               50200               50200
## 5      5           57500               57500               57400
## 6      6           67200               67200               67200
## 7      7           80900               81400               81300
## 8      8          102900              102800              102800
## 9      9          128200              128500              128400
## 10    10          162000              161500              161700
## 11    11          194100              195300              195200
## 12    12          238100              239800              240000
```

As the constant range width versus varied range width reviews occurred on the prior structures and the personal preference for varied, for these three new structures, simply will apply the varied range width.

First - Reduced structure

```
MktStruc_Reduc_Rnd_Var_RW<-as.data.frame(Market_Midpoints_Reduc_Rnd[,c(3)])
colnames(MktStruc_Reduc_Rnd_Var_RW)[1]<-"Midpoint"
MktStruc_Reduc_Rnd_Var_RW = MktStruc_Reduc_Rnd_Var_RW%>%
  mutate(Minimum = ifelse(as.integer(Grade_Market)<5, Midpoint/(1+(.5*MktRng1)),
                          ifelse(as.integer(Grade_Market)<9,Midpoint/(1+(.5*MktRng2)),
                                 Midpoint/(1+(.5*MktRng3)))))
MktStruc_Reduc_Rnd_Var_RW<-mround(MktStruc_Reduc_Rnd_Var_RW,50)
MktStruc_Reduc_Rnd_Var_RW$Maximum<-MktStruc_Reduc_Rnd_Var_RW$Midpoint-
  MktStruc_Reduc_Rnd_Var_RW$Minimum+MktStruc_Reduc_Rnd_Var_RW$Midpoint
MktStruc_Reduc_Rnd_Var_RW<-MktStruc_Reduc_Rnd_Var_RW[,c(2,1,3)]
MktStruc_Reduc_Rnd_Var_RW<-cbind(Grade_Market,MktStruc_Reduc_Rnd_Var_RW)
colnames(MktStruc_Reduc_Rnd_Var_RW)[1]<-"Grade"
MktStruc_Reduc_Rnd_Var_RW$MinCR<-round(MktStruc_Reduc_Rnd_Var_RW$Minimum/
                                           MktStruc_Reduc_Rnd_Var_RW$Midpoint,4)
MktStruc_Reduc_Rnd_Var_RW$MaxCR<-round(MktStruc_Reduc_Rnd_Var_RW$Maximum/
                                           MktStruc_Reduc_Rnd_Var_RW$Midpoint,4)
MktStruc_Reduc_Rnd_Var_RW$Structure<-"Reduced"
MktStruc_Reduc_Rnd_Var_RW
```

```
##    Grade Minimum Midpoint Maximum  MinCR  MaxCR Structure
## 1      1   20650    25300   29950 0.8162 1.1838   Reduced
## 2      2   23250    28500   33750 0.8158 1.1842   Reduced
## 3      3   26350    32300   38250 0.8158 1.1842   Reduced
## 4      4   30600    37500   44400 0.8160 1.1840   Reduced
## 5      5   33300    43300   53300 0.7691 1.2309   Reduced
## 6      6   39000    50700   62400 0.7692 1.2308   Reduced
## 7      7   47300    61500   75700 0.7691 1.2309   Reduced
## 8      8   58850    76500   94150 0.7693 1.2307   Reduced
## 9      9   69950    96200  122450 0.7271 1.2729   Reduced
## 10    10   88000   121000  154000 0.7273 1.2727   Reduced
## 11    11  105250   144700  184150 0.7274 1.2726   Reduced
## 12    12  129000   177400  225800 0.7272 1.2728   Reduced
```

Next is the baseline structure

```
MktStruc_BLine_Rnd_Var_RW<-as.data.frame(Market_Midpoints_BLine_Rnd[,c(3)])
colnames(MktStruc_BLine_Rnd_Var_RW)[1]<-"Midpoint"
MktStruc_BLine_Rnd_Var_RW = MktStruc_BLine_Rnd_Var_RW%>%
  mutate(Minimum = ifelse(as.integer(Grade_Market)<5, Midpoint/(1+(.5*MktRng1)),
                          ifelse(as.integer(Grade_Market)<9,Midpoint/(1+(.5*MktRng2)),
                                 Midpoint/(1+(.5*MktRng3)))))

MktStruc_BLine_Rnd_Var_RW<-mround(MktStruc_BLine_Rnd_Var_RW,50)
MktStruc_BLine_Rnd_Var_RW$Maximum<-MktStruc_BLine_Rnd_Var_RW$Midpoint-
  MktStruc_BLine_Rnd_Var_RW$Minimum+MktStruc_BLine_Rnd_Var_RW$Midpoint
MktStruc_BLine_Rnd_Var_RW<-MktStruc_BLine_Rnd_Var_RW[,c(2,1,3)]
MktStruc_BLine_Rnd_Var_RW<-cbind(Grade_Market,MktStruc_BLine_Rnd_Var_RW)
colnames(MktStruc_BLine_Rnd_Var_RW)[1]<-"Grade"
MktStruc_BLine_Rnd_Var_RW$MinCR<-round(MktStruc_BLine_Rnd_Var_RW$Minimum/
```

```
                                          MktStruc_BLine_Rnd_Var_RW$Midpoint,4)
MktStruc_BLine_Rnd_Var_RW$MaxCR<-round(MktStruc_BLine_Rnd_Var_RW$Maximum/
                                          MktStruc_BLine_Rnd_Var_RW$Midpoint,4)
MktStruc_BLine_Rnd_Var_RW$Structure<-"Baseline"
MktStruc_BLine_Rnd_Var_RW
```

```
##    Grade Minimum Midpoint Maximum  MinCR  MaxCR Structure
## 1      1   23750    29100   34450 0.8162 1.1838  Baseline
## 2      2   26800    32800   38800 0.8171 1.1829  Baseline
## 3      3   30600    37500   44400 0.8160 1.1840  Baseline
## 4      4   35200    43100   51000 0.8167 1.1833  Baseline
## 5      5   38300    49800   61300 0.7691 1.2309  Baseline
## 6      6   44450    57800   71150 0.7690 1.2310  Baseline
## 7      7   53900    70100   86300 0.7689 1.2311  Baseline
## 8      8   67750    88100  108450 0.7690 1.2310  Baseline
## 9      9   80750   111000  141250 0.7275 1.2725  Baseline
## 10    10  101900   140100  178300 0.7273 1.2727  Baseline
## 11    11  121150   166600  212050 0.7272 1.2728  Baseline
## 12    12  151850   208800  265750 0.7273 1.2727  Baseline
```

Last is the premium structure

```
MktStruc_Prem_Rnd_Var_RW<-as.data.frame(Market_Midpoints_Prem_Rnd[,c(3)])
colnames(MktStruc_Prem_Rnd_Var_RW)[1]<-"Midpoint"
MktStruc_Prem_Rnd_Var_RW = MktStruc_Prem_Rnd_Var_RW%>%
  mutate(Minimum = ifelse(as.integer(Grade_Market)<5, Midpoint/(1+(.5*MktRng1)),
                          ifelse(as.integer(Grade_Market)<9,Midpoint/(1+(.5*MktRng2)),
                                 Midpoint/(1+(.5*MktRng3)))))
MktStruc_Prem_Rnd_Var_RW<-mround(MktStruc_Prem_Rnd_Var_RW,50)
MktStruc_Prem_Rnd_Var_RW$Maximum<-MktStruc_Prem_Rnd_Var_RW$Midpoint-
  MktStruc_Prem_Rnd_Var_RW$Minimum+MktStruc_Prem_Rnd_Var_RW$Midpoint
MktStruc_Prem_Rnd_Var_RW<-MktStruc_Prem_Rnd_Var_RW[,c(2,1,3)]
MktStruc_Prem_Rnd_Var_RW<-cbind(Grade_Market,MktStruc_Prem_Rnd_Var_RW)
colnames(MktStruc_Prem_Rnd_Var_RW)[1]<-"Grade"
MktStruc_Prem_Rnd_Var_RW$MinCR<-round(MktStruc_Prem_Rnd_Var_RW$Minimum/
                                              MktStruc_Prem_Rnd_Var_RW$Midpoint,4)
MktStruc_Prem_Rnd_Var_RW$MaxCR<-round(MktStruc_Prem_Rnd_Var_RW$Maximum/
                                              MktStruc_Prem_Rnd_Var_RW$Midpoint,4)
MktStruc_Prem_Rnd_Var_RW$Structure<-"Premium"
MktStruc_Prem_Rnd_Var_RW
```

```
##    Grade Minimum Midpoint Maximum  MinCR  MaxCR Structure
## 1      1   27000    33100   39200 0.8157 1.1843   Premium
## 2      2   31000    38000   45000 0.8158 1.1842   Premium
## 3      3   35350    43300   51250 0.8164 1.1836   Premium
## 4      4   41000    50200   59400 0.8167 1.1833   Premium
## 5      5   44250    57500   70750 0.7696 1.2304   Premium
## 6      6   51700    67200   82700 0.7693 1.2307   Premium
## 7      7   62600    81400  100200 0.7690 1.2310   Premium
## 8      8   79100   102800  126500 0.7695 1.2305   Premium
## 9      9   93450   128500  163550 0.7272 1.2728   Premium
## 10    10  117450   161500  205550 0.7272 1.2728   Premium
```

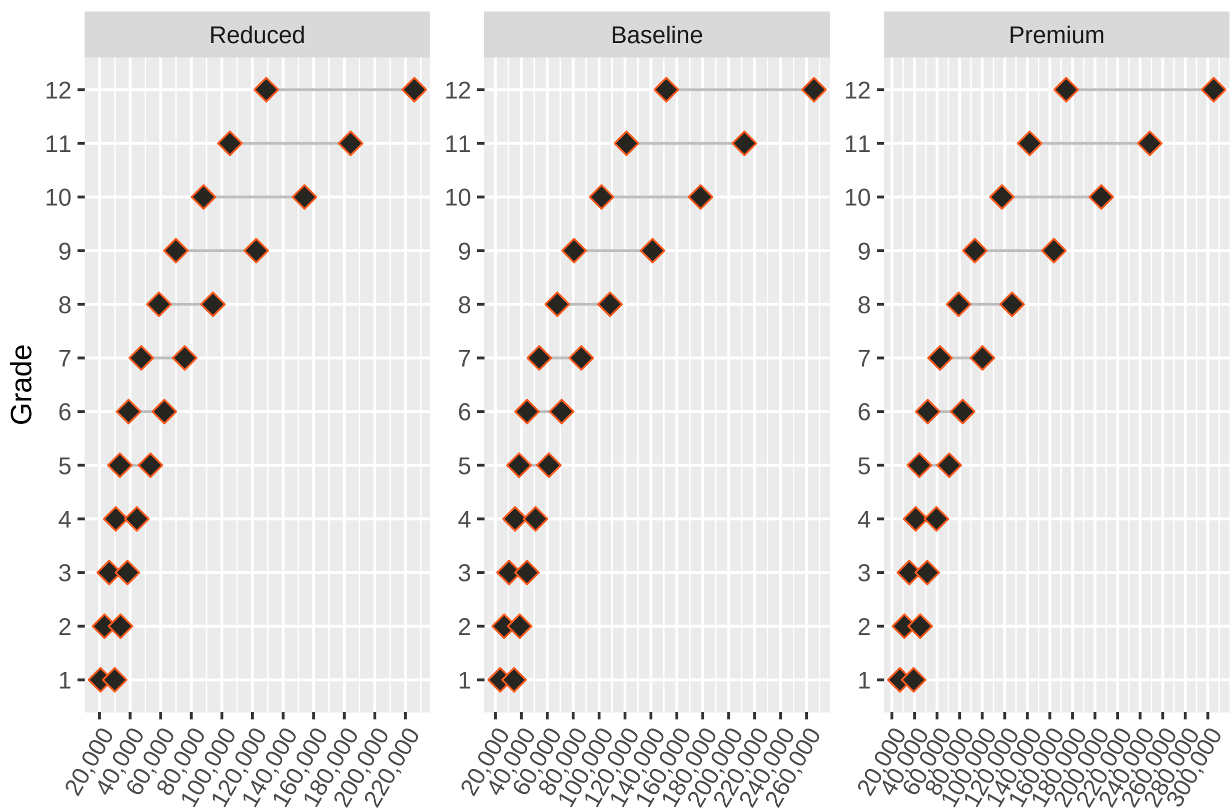
```
## 11     11  142050    195300   248550 0.7273 1.2727    Premium
## 12     12  174400    239800   305200 0.7273 1.2727    Premium
```

Now that the three structures are created, it may be logical to combine all three into a single data frame.

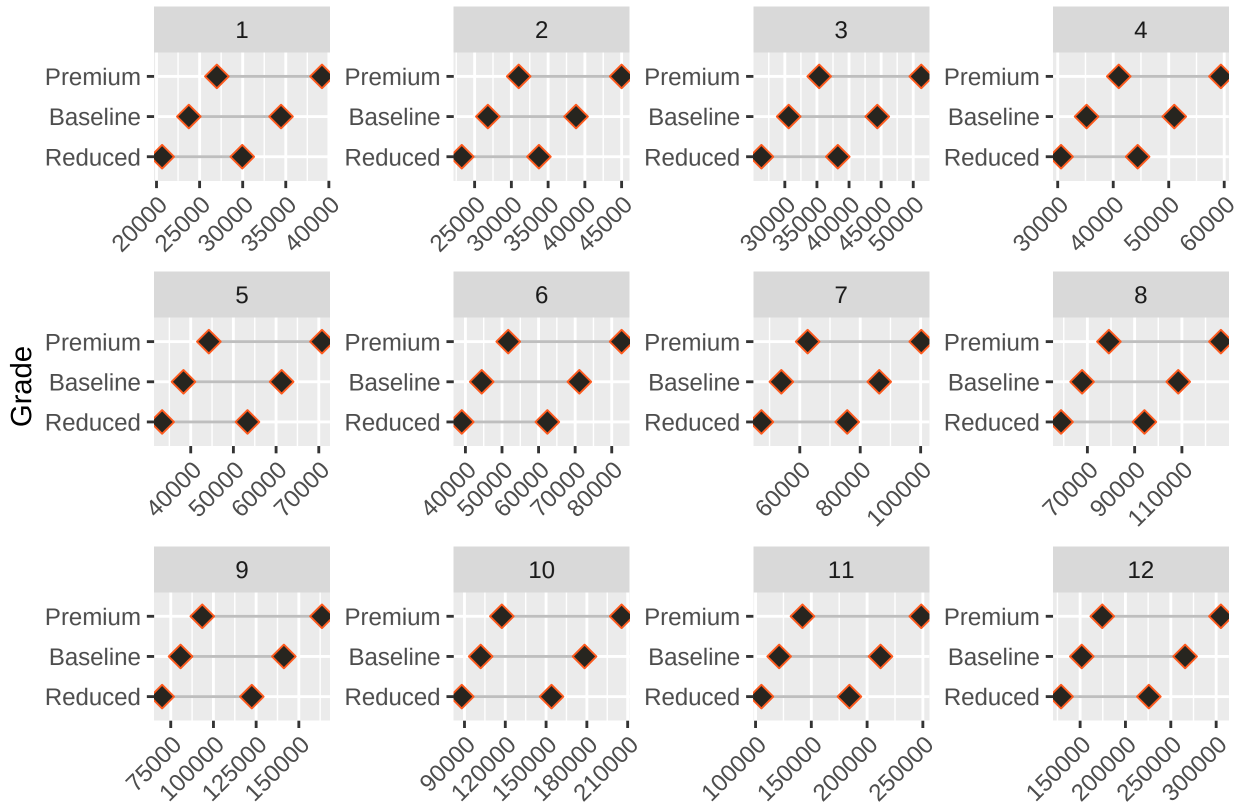
```
Mrk_Final_RangeWidth<-rbind(MktStruc_Reduc_Rnd_Var_RW,MktStruc_BLine_Rnd_Var_RW,
                                                    MktStruc_Prem_Rnd_Var_RW)
attach(Mrk_Final_RangeWidth)
Mrk_Final_RangeWidth$Structure<-factor(Structure, levels=c("Reduced","Baseline",
                                                                      "Premium"))
detach()
Mrk_Final_RangeWidth
```

Concluding the design of these structures, we can view how all three structures look visually, however in 2 different ways; *By Structure* By Grade

```
ggplot(Mrk_Final_RangeWidth) +
  facet_wrap(~Structure, ncol = 3, scales = "free")+
  geom_segment(aes(x=Grade, xend=Grade, y=Minimum, yend=Maximum), color="grey")+
  geom_point(aes(x=Grade, y=Minimum ),color="#FD5A1E",size=3,shape=23,fill="#27251F")+
  geom_point(aes(x=Grade, y=Maximum ),color="#FD5A1E",size=3,shape=23,fill="#27251F")+
  coord_flip()+
  ylab("")+
  xlab("Grade")+
  scale_y_continuous(label=comma, breaks=seq(0,350000,20000))+
  theme(axis.text.x = element_text(angle = 60, vjust = 1, hjust=1))
```



```
ggplot(Mrk_Final_RangeWidth) +
  facet_wrap(~Grade, ncol = 4, scales = "free")+
  geom_segment(aes(x=Structure, xend=Structure, y=Minimum, yend=Maximum), color="grey")+
  geom_point(aes(x=Structure, y=Minimum ),color="#FD5A1E",size=3,shape=23,fill="#27251F")+
  geom_point(aes(x=Structure, y=Maximum ),color="#FD5A1E",size=3,shape=23,fill="#27251F")+
  coord_flip()+
  ylab("")+
  xlab("Grade")+
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))
```



“‘