Executive Summary

My father, Robert owns a Baskin Robbins franchise in Scotland. He is always looking for new ways to improve the business, and his income. The store is small, but has a prime location close to the city center. Most of the day-time customers are passing the store on foot (usually returning from town). The night-time customers almost always drive to the store. The store currently has 10 employees, and most of them are students.

The weather in Scotland is very temperamental, and it plays a big part in how much business the store generates. Typically, demand goes down during inclement weather, but the labor supply stays the same. Also, when it is unusually hot, large queues will form and customers become impatient and sometime leave. The problem I hope to remedy is over and under-staffing. If Robert could better predict demand for ice-cream, he could avoid over and under-staffing.

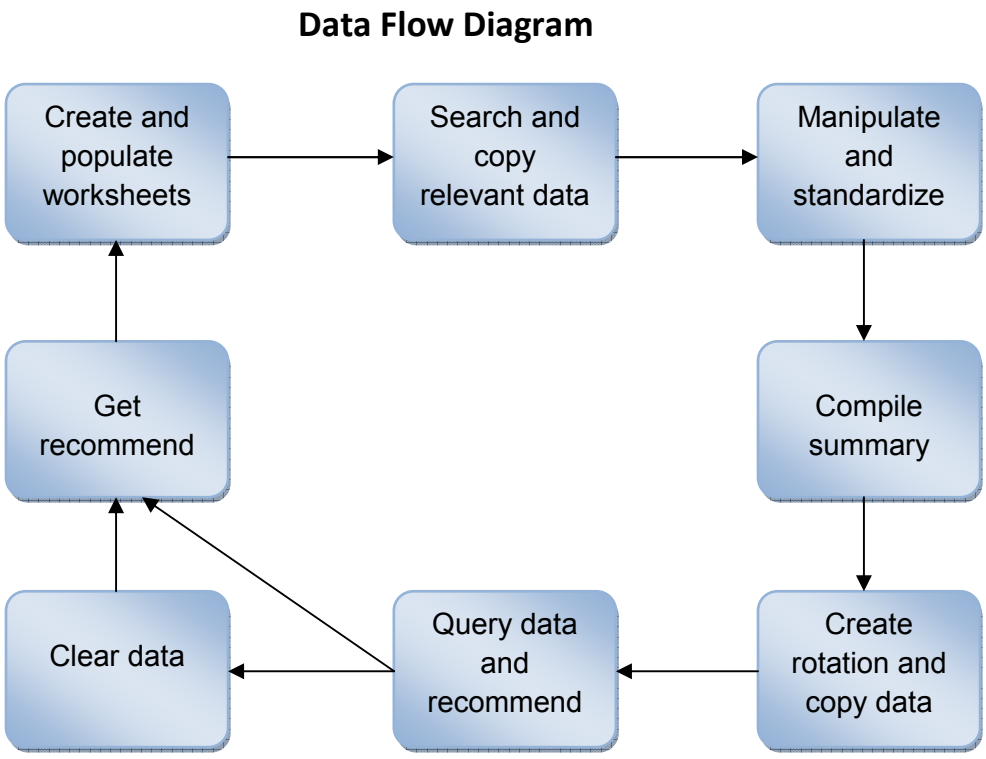
Robert does periodically check the weather, but he does not gather enough information to persuade him to adjust the staff rotation.

The VBA program which I designed pulls together weather data from multiple web sources. The program standardizes the data and compiles a weather summary. This summary averages out the forecast temperatures and chance of precipitation from the different web sources. The summary is then translated into staffing recommendations. These recommendations are obtained using thresholds for varying weather conditions. The user will have the option of viewing a 5-day forecast and also a 24-hour forecast.

Robert now has a toll that can improve the efficiency of the store, improve his profits, and take a process that would have taken him 20 minutes down to a few seconds.

Process flow

To make the construction of this program simpler, it was important to have the data flow through stages as opposed to jumping from the website straight to the rotation (see diagram below). First, the program creates temporary storage worksheets in which to import the website data. After populating the worksheets, the program then searches and copies the relevant data from each temporary worksheet to a single permanent worksheet. The data are then manipulated and standardized, and become ready to compile into a summary. The data are compiled into a summary which is readable by the user. A copy of the existing rotation is made and then some of the summary data is copied over to this new rotation. Finally, the new rotation runs queries based on a combination of the data on the new rotation and data on from the summary worksheet to produce recommendations. Buttons were added to the original rotation by which the user could get recommendations or clear all data and new rotations.



Web Sources, Importing and Searching

To begin writing this program I first had to find web sources containing weather forecasts for Aberdeen, Scotland. The sources I used include the BBC, the Met Office, MSN, Time and Date, AccuWeather, and Scotland TV. Not all sources had all the information I needed (i.e. outlook, day temp, night temp, chance of precipitation) but collectively the composite data was sufficient.

After finding the URLs I needed, I programmed VBA to create temporary storage worksheets for each URL, and then to import all data from each URL. This import gave me excess data that I did not want, but it was necessary. If I had imported just the boxes I needed, any future additions of boxes to the web page would have left me with the wrong data had I tried to import again. Searching for a value that was unique and then using a relative reference from that value gave me the access to the weather data I needed—day, time, outlook, day temperature, night temperature, hourly temperature, and chance of precipitation. I had some difficulty with this, but I will explain that in a later section.

After locating the relevant data, I copied them to a "summary" worksheet. Each different website's data are copied to this summary worksheet. The summary worksheet already exists in the file. I thought it would be best to have this worksheet be permanent, so the user can look back at the data where the recommendations come from. It was in this summary worksheet that I did a lot of data manipulation and standardizing.

Summary Worksheet

Summary Data

	Fri	Sat	Sun	Mon	Tue
Description (Accu)	Partly Sunny	Partly Sunny	Intermittent Clouds	Intermittent Clouds	
Description (BBC)	White Cloud	Sunny Intervals	Sunny Intervals	Sunny Intervals	White Cloud
Description (T&D)		Sprinkles late. More sun than clouds. Cool.	More sun than clouds. Cool.	Sprinkles early. Afternoon clouds. Cool.	Mostly cloudy. Cool.
Day High (°C)	8	7.2	5.2	3.5	-1.0
Night Low (°C)	1.5	2.3	2.4	-3.0	1.3
Average Temperature (°	3	4.8	3.8	2.6	2.3
Chance of precipitation	35%	23%	23%	10%	15%

Individual Source Data

AccuWeather:					
Description (Accu)	Partly Sunny	Partly Sunny	Intermittent Clouds	Intermittent Clouds	
Day High (°C)	8	8	3	3	
Night Low (°C)	2	1	1	1	
Average Temperature (°C)	5	4.5	2	2	
BBC:					
Description (BBC)	White Cloud	Sunny Intervals	Sunny Intervals	Sunny Intervals	White Cloud
Day High (°C)		7	7	3	2
Night Low (°C)	1	2	3	2	1
Average Temperature (°C)	1	4.5	5	2.5	1.5
MSN:					
Chance of precipitation	35%	25%	15%	5%	20%
Time and Date:					
Description (T&D)		Sprinkles late. More sun than clouds. Cool.	More sun than clouds. Cool.	Sprinkles early. Afternoon clouds. Cool.	Mostly cloudy. Cool.
Day High (°C)		6.7	5.6	4.4	4.4
Night Low (°C)		3.9	3.3	2.2	1.7
Average Temperature (°C)		5.3	4.4	3.3	3.1
Chance of precipitation		20%	30%	15%	10%
	Celcius Conversion				
	44	42	40	40	
	39	38	36	35	

Data Manipulation and Standardization

The following adjustments had to be made to make the data ready to be compiled into the summary:

- One website had the temperature in Fahrenheit so I had to change it to Celsius.
- The MSN forecast description (located on the 24-hour summary sheet) was repeating itself. For example “Sunny” would read “SunnySunny”. I used a “Do Loop” that basically cut the text in half to fix the problem. The loop is as follows:

```
x = ActiveCell.Column

Do Until Cells(21, x).Value = ""
    y = Len(Cells(21, x).Value)
    y = y / 2
    Cells(21, x).Value = Mid(Cells(21, x).Value, 1, y)

    x = x + 1

Loop
```

- The temperature values had to appear as a number only. All but one of the sources had characters I could not work with, such cases were similar to this: “high: 3°C”. To remove the unwanted characters I used the following “If” statements:

```
If Len(ActiveCell) = 9 Then
    x = 1
End If
If Len(ActiveCell) = 10 Then
    x = 2
End If
If Len(ActiveCell) = 11 Then
    x = 3
End If

ActiveCell.Value = Mid(ActiveCell, 6, x)
```

After the data was cleaned and hidden, I compiled it into a summary on the same permanent worksheet (see Summary Worksheet insert above). The ‘Average’ formulas were used to average the temperatures and the chances of precipitation. As part of the summary, there is a loop that highlights the temperatures that are particularly extreme. If a temperature is higher than 24 degrees Celsius during the day, or 22 degrees Celsius during the night, then the cell with the temperature will be highlighted red. If a temperature is lower than negative 1 during the day, or negative 3 during the night, then the cell with the temperature will be highlighted blue. The code is as follows:

```
f = 2

Do Until Cells(7, f).Value = ""

Cells(7, f).Select
If Cells(7, f).Value >= 24 Then

With Selection.Interior
    .Pattern = xlSolid
    .PatternColorIndex = xlAutomatic
    .Color = 255
    .TintAndShade = 0
    .PatternTintAndShade = 0
End With
End If

f = f + 1

Loop

f = 2
```

Preparing for the Recommendation

After manipulating, standardizing, and organizing the website data, and creating a summary table that compiles all of the data, a recommendation is produced. The recommendation is the whole purpose of this program, so it was important to get it right.

The program creates a new worksheet with an exact copy of the existing rotation. It then pulls the day or time (for the 24-hour recommend) from the summary worksheet and copies it into the new rotation worksheet. I wanted to give the user as little to look at as I could on this worksheet, so it would be easier to read. After all, the user can always look at the more detailed data by clicking on the summary worksheet. For the 24-hour recommend, it was important to determine whether the time occurred during the day, night, or not even during hours of business (the 5-day forecast had that information built into the data as I was extracting day temperatures and night temperatures from the websites). I needed to determine this because my recommendations were partly determined based on the whether it was day or night. I wrote a loop that searched through the times and labeled the cell next to it “Day”, “Night”, or “Store closed” depending on what the time was. The heart of the program is next—figuring out what staffing recommendation should be given after taking the summary data into consideration.

The Recommendation

The recommendation is formulated largely by “If” statements and nested “If” statements. The parameters are narrowed for each weather circumstance until the appropriate recommendation can be found. For example, if it is nighttime (I previously determined “Night” as any time after 6pm), and the

temperature is between 13 Celsius and 21 Celsius, and there is less than a 50 percent chance of precipitation, then the program will recommend staffing four employees. However, if it is daytime, and the temperature is between 15°C and 22°C, and there is more than a 65 percent chance of precipitation, then the program will recommend staffing one employee. The following code is from the 24-hour sub procedure and would produce such a recommendation.

```
If Cells(x, 4).Value = "Day" Then

'If it is between 15 and 22 degrees
    If Cells(x, 2).Value >= 15 Then
        If Cells(x, 2).Value < 22 Then
            If Cells(x, 3).Value > 0.65 Then
                Cells(x, 6).Value = "Put 1 person on"
            Else
                Cells(x, 6).Value = "Put 2 people on"
            End If
        End If
    End If
```

The staffing recommendation is based on various scenarios like the one above. I worked at the store for over 2 years and am very familiar with the fluctuations in demand due to the weather. However, as these are only recommendations, the program does not alter anything on the existing rotation. Decisions to alter will be made by management and will be based on employee availability. A screenshot of the existing rotation and the 24-hour recommendations is on the next page.

Baskin Robbins Staff Rota

Week Ending 3rd Octber 2010

	Monday		Tuesday		Wednesday		Thursday		Friday		Saturday		Sunday	
	27		*28*		*29*		*30*		*1*		*2*		*3*	
Hannah M	18:00	22:45							18:00	23:15			13:45	22:45
Caroline			18:00	22:45										
Gail							18:00	22:45			18:00	23:15		
Kate					18:00	22:45	11:15	18:00						
Hannah G			11:15	18:00					14:00	18:00	19:00	23:15	13:45	18:00
Lara	11:15	18:00			11:15	18:00	18:00	22:45	11:15	18:00			18:00	22:45
Jessica							13:00	16:00			18:00	23:15		
Fiona					18:00	22:45							18:00	22:45
Rhona	11:15	18:00	18:00	22:45					18:00	23:15	11:15	18:00		
Kelly	18:00	22:45	13:00	16:00					19:00	23:15	14:00	18:00		

Recommendations:

600	Store closed	NA
900	Store closed	NA
1200	Day	Put a monkey on
1500	Day	Put 1 person on
1800	Night	Put 2 people on
2100	Night	Put 2 people on
0	Store closed	NA
300	Store closed	NA

Please Note

- There will always be at least two workers at night for security reasons.
- The recommendation, “put a monkey on” means that the store will be so quiet that a monkey could run it.
- The program does not take holidays into account.
- This program is most helpful when there is more inconsistent weather.

Conclusion

This program is very usable. Though Robert user may not swear by it (after all the recommendation is only as good as the web sources, and they have been known to be wrong) or use it as the ultimate staffing scheduler, he will still use it. Why? Because it is an excellent starting place for staffing, it is a second opinion, it provides guidance, and it provides a multiple source weather forecast from six sources at the click of a button.

Directions for use

The program could not be simpler to use. There are four buttons that have macros assigned to them on the original rotation worksheet. The buttons have the following functions:

- First button runs the 5-day forecast and produces a new rotation with recommendation
- Second button clears all the 5-day forecast data and deletes the new rotation with recommendation
- Third button runs the 24-hour forecast and produces a new rotation with recommendation
- Fourth button clears all the 24-day forecast data and deletes the new rotation with recommendation

If a forecast and new rotation with a recommendation already exist, clicking the “Get Forecast” button will replace the existing forecast and recommendation. If the “Clear Data” button is clicked and there is no data to clear or new rotation to delete, nothing will happen (at least that the user will see).

Difficulties Encountered

I encountered a lot more difficulties writing this program than I thought I would. One of my major sources of frustration was the lack of uniformity between the web sites. You will notice that there are sometimes gaps in the individual source data and the summary data. These gaps are caused by an updating lag between the websites (some update before others). I have discovered that most of the lags are late at night (Scotland time), so this should not be a major problem for the user.

The 24-hour forecast import was particularly hard to handle because the data I needed would move around rendering relative reference useless. I discovered that a “Do Loop” would solve the problem. I used the loop to delete rows (they were always blank rows) that appeared between the cells I wanted. What the loop effectively did was organize my data cells one after the other; I could then use relative reference. I had to perform the loop twice because there were often 2 rows one after the other in between my data. If I deleted row “164” because it was blank my next row up for evaluation would be row “164 + 1”, or row “165”. However, row “165” was not the row I wanted to look at next; I wanted to look at the new row “164”. Row “165” was previously known as row “166” before I deleted row “164”. So I would actually miss a row by doing this loop. I watched my code run through the whole spreadsheet a few times before I realized what was going on. The code I wrote can be seen on the following page.

```
ActiveCell.Offset(1, 1).Range("A1").Select

x = ActiveCell.Row

Do Until Cells(x, 2).Value = "Stop Here"
    If Cells(x, 2).Value = "" Then
        Cells(x, 2).Select
        Selection.EntireRow.delete
    End If

    x = x + 1

Loop
```

The program does not take holidays into consideration; this is one element I wanted to include. However the data was far too inconvenient and disorganized to make this a possibility. I feel like if I continued on the holiday inclusive path I started on it was going to at least double the length of the project.

It took me a while to find decent websites. I ended up discarding three websites that I had originally planned on using. I have learned that extracting form websites is risky business. One issue I encountered was a web page that could change from 5-day forecasts to 24-hour forecasts without changing the URL. I kept getting inconsistent data from a couple of websites for the 24-hour forecast, and I finally realized it was because it was giving me 5-day data; the URL initiated on the 5-day forecast.