Use JSL to Scrape Data from the Web and Predict Football Wins!

University of New Hampshire

William Baum
Graduate Statistics Student
University of New Hampshire

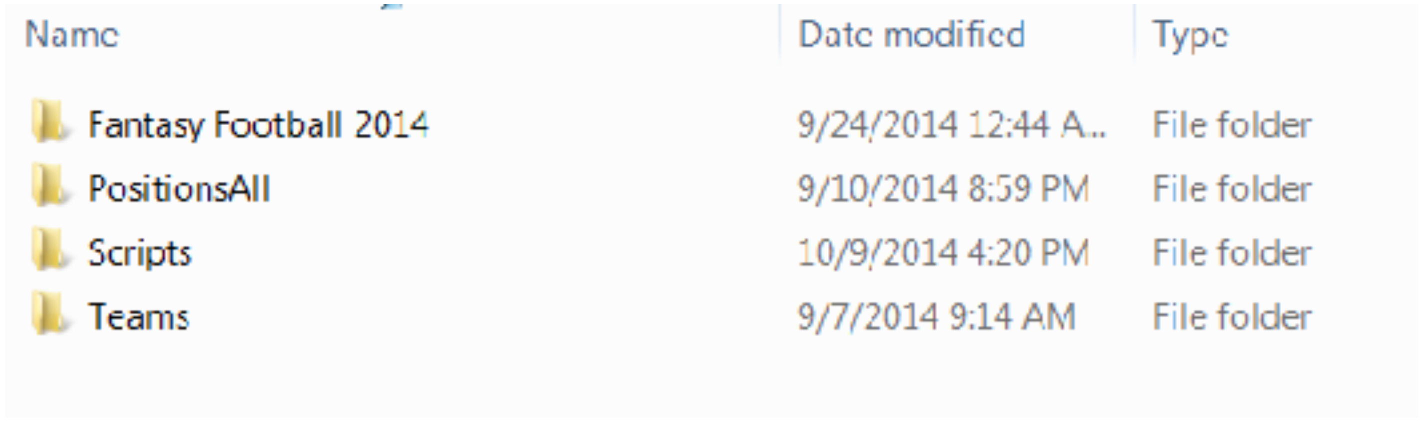
Just for Fun!

- I’m an avid American football fan
- Sports statistics are easily accessible, online
- My goal: to predict total wins in a season, by team
- These techniques can be applied to any web data
- It’s fun to see how JMP Scripting Language [JSL] can, at once, automate data clean up and perform modeling
- With a little JSL code this really is easy!
  - JMP actually helps you write the code through platforms
  - It’s straightforward and can be quite powerful
  - Saves a lot of time when updating models with new data

JSL Resources

- As a newcomer to JMP Scripting Language [JSL]
  - I did not always know the best place to turn for answers
  - Many resources exist
- JMP online community: JMP website & online forums
- Excellent resources are available from JMP:
  - Help Menu > JSL functions
  - Online Scripting Guide & Syntax Reference
    - http://www.jmp.com/support/help/Scripting_Guide.shtml
    - http://www.jmp.com/support/help/JSL_Syntax_Reference.shtml
  - Books that will save you loads of time:
    - **Jump Into JMP Scripting**, by Wendy Murphrey and Mary Lucas
    - **JSL Companion**, by Theresa Utlaut, Georgia Morgan, & Kevin Anderson

Preparation



Name	Date modified	Type
Fantasy Football 2014	9/24/2014 12:44 A...	File folder
PositionsAll	9/10/2014 8:59 PM	File folder
Scripts	10/9/2014 4:20 PM	File folder
Teams	9/7/2014 9:14 AM	File folder

- Create folders to store files:
  - Any required directories or subdirectories
  - Good way to organize files created from different data sources
  - Directory names will be used in JSL scripts

Scrape Data: Team Statistics

Offense, Defense, Special Teams

1. Set local path for file storage:

- dir = Pick Directory(“Select directory for file storage”, show files(0));

2. Get data from a website & save the file (one for each HTML table):

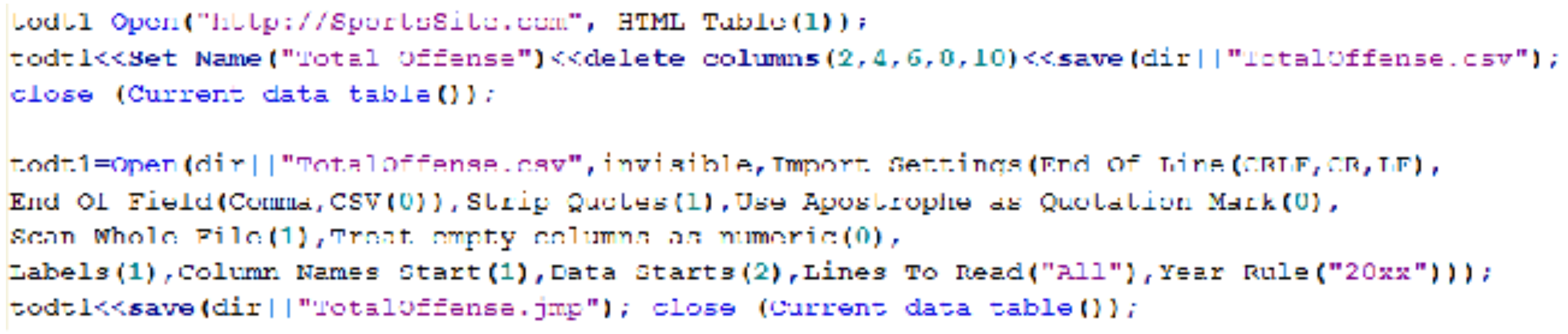
- ObjectName=Open(“http//:url_name1.com”, HTML Table(#)) << Set Name(“Name”) << Save(dir||“filename.csv”);

- Simply copy and paste URL names into your code from your browser
- To put a link to the website directly in your script helps to open webpage while editing code: Web("http://url_name.com");

3. Why save as CSV file first?

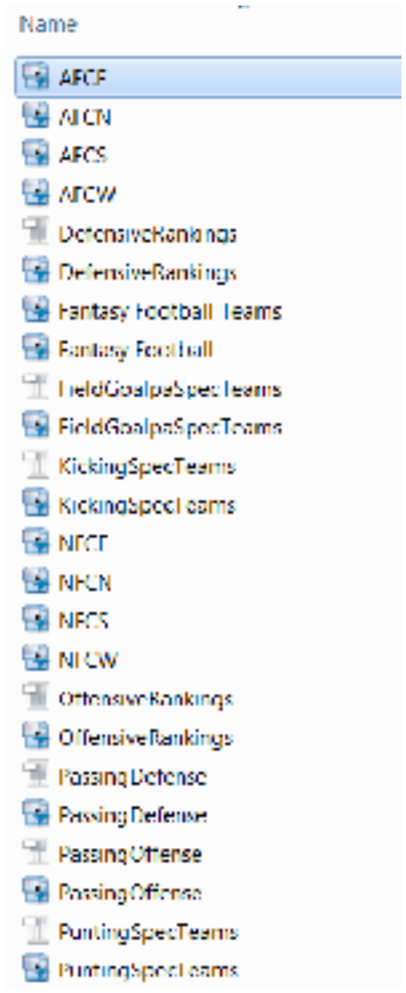
- Data Import Options, to support automated data import:
  - Import Settings(...), Labels(1), Column Names Start(2), Data Starts(3), etc.
- Delete empty columns! Can also use For loop to eliminate them.

The Code: Scrape Data



```
todt1 Open("http://SportsSite.com", HTML Table(1));
todt1<<Set Name("Total Offense")<<delete columns(2,4,6,8,10)<<save(dir||"TotalOffense.csv");
close (Current data table());

todt1=Open(dir||"TotalOffense.csv",invisible,Import Settings(End Of Line(CRLF,CR,LF),
End Of Field(Comma,CSV(0)),Strip Quotes(1),Use Apostrophe as Quotation Mark(0),
Scan Whole File(1),Treat empty columns as numeric(0),
Labels(1),Column Names Start(1),Data Starts(2),Lines To Read("All"),Year Rule("20xx")));
todt1<<save(dir||"TotalOffense.jmp"); close (Current data table());
```

The Files



Name

AFCE
AFCN
AFCS
AFCW
DefensiveRankings
DefensiveRankings
Fantasy Football Teams
Fantasy Football
FieldGoalpaSpecTeams
FieldGoalpaSpecTeams
KickingSpecTeams
KickingSpecTeams
NFCE
NFCN
NFCS
NFCW
OffensiveRankings
OffensiveRankings
PassingDefense
PassingDefense
PassingOffense
PassingOffense
PuntingSpecTeams
PuntingSpecTeams

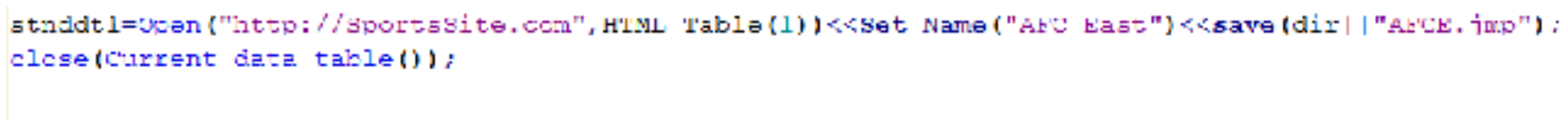
- JMP will now automatically pull the information from the web and save the files.
- Testing may be required!
- No need to save these files, but:
  - Data can be a bit messy
  - Easier for debugging
  - One can see the imported tables and adjust the code

Standings Data
(Wins, Losses, etc.)

Get additional data from another website

- Again, using a little object-oriented scripting
- This time, the data do not require any pre-processing to parse correctly – Nice!
- Imported directly into JMP and saved

The Code: Standings Data

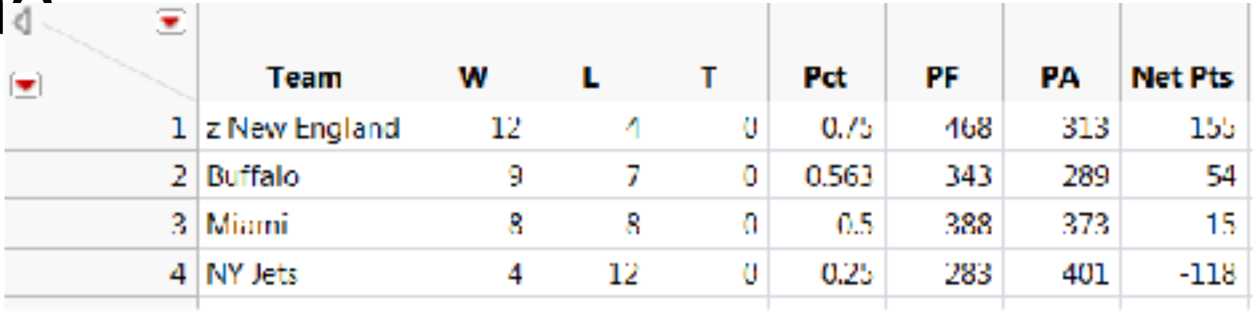


```
stnddt1=Open("http://SportsSite.com",HTML Table(1))<<Set Name("AFC East")<<save(dir||"AFCE.jmp");
close(Current data table());
```

ObjectName = Open("http://url_name2", HTML Table(#)) << Set Name("TableName") << Save(dir||"FileName.jmp")

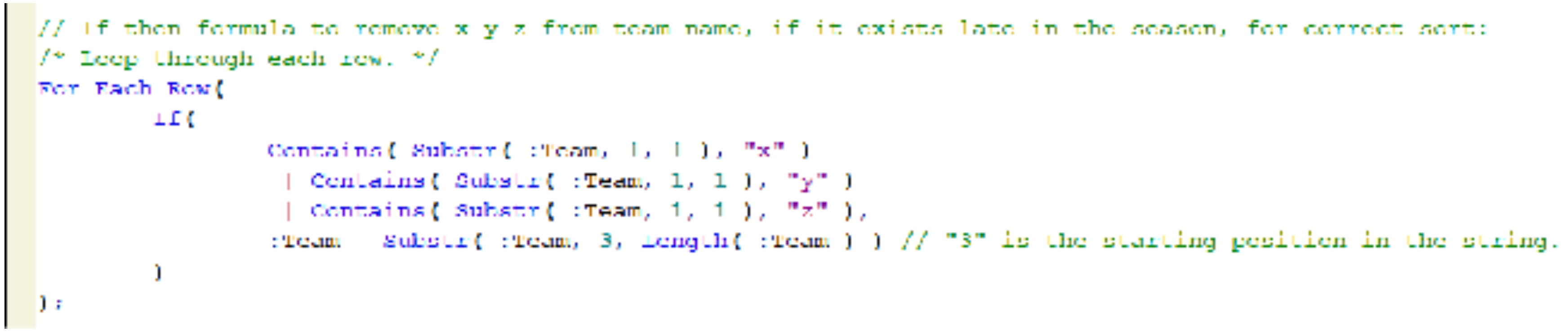
Cell Processing

- Sometimes even slight variations in individual strings can foul up your results
- In this case, the playoffs are near…
- To indicate New England has clinched a division title, a “z” is inserted into their name



	Team	W	L	T	Pct	PF	PA	Net Pts
1	z New England	12	4	0	0.75	468	313	155
2	Buffalo	9	7	0	0.563	343	289	54
3	Miami	8	8	0	0.5	388	373	15
4	NY Jets	4	12	0	0.25	283	401	-118

The Code: Cell Processing



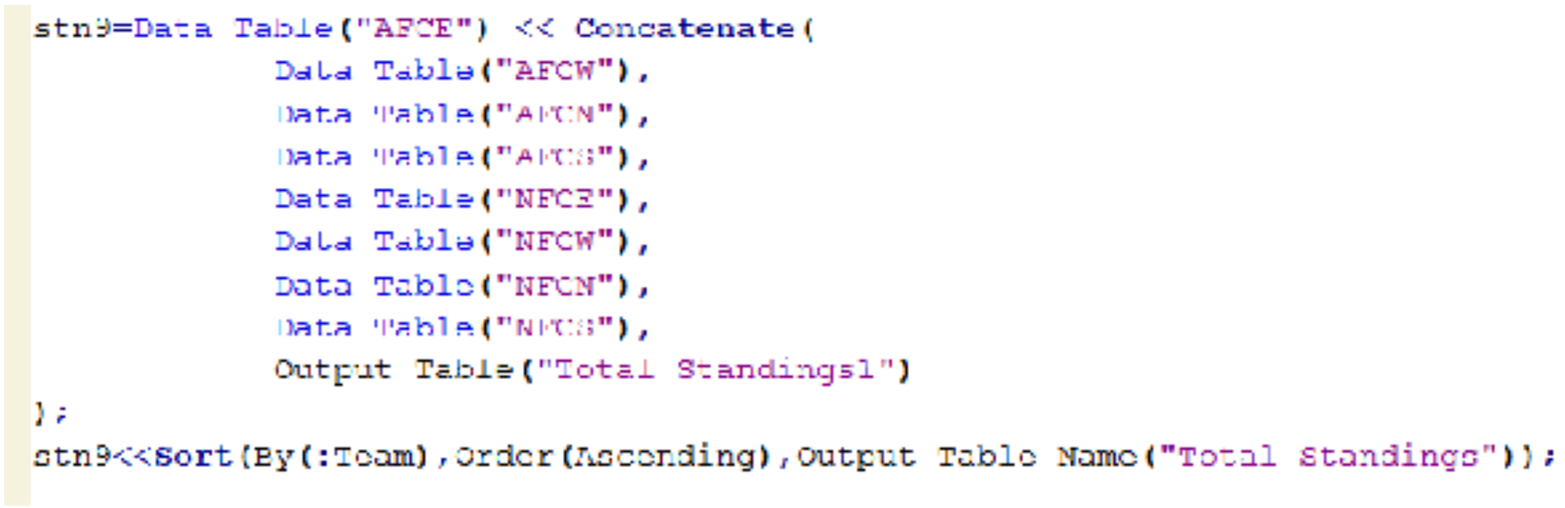
```
// If then formula to remove x y z from team name, if it exists late in the season, for correct sort:
/* Loop through each row. */
For Each Row(
        If(
                Contains( Substr( :Team, 1, 1 ), "x" )
                 | Contains( Substr( :Team, 1, 1 ), "y" )
                 | Contains( Substr( :Team, 1, 1 ), "z" ),
                :Team = Substr( :Team, 3, Length( :Team ) ) // "3" is the starting position in the string.
        )
);
```

- Cell processing to the rescue!
- JMP loops through each row of the Team name column
- For Each Row command makes for easy looping
- Conditional “If, then” phrase
- Respectively, “Contains” and “Substr” commands:
  - returns the position of a specified item within a string
  - returns part of a string: starting at a specified position, returns a specified number of characters

Reopen Files, Concatenate & Sort

- Reopen the files, invisibly
- Concatenate – different rows, same columns
- Sort the rows of the resultant file by team name, alphabetically, to ensure like rows are matched during the joining process, later on
- Save the new table, which contains the aggregate information
- Join – different columns, same row labels

The Code: Open, Concatenate & Sort

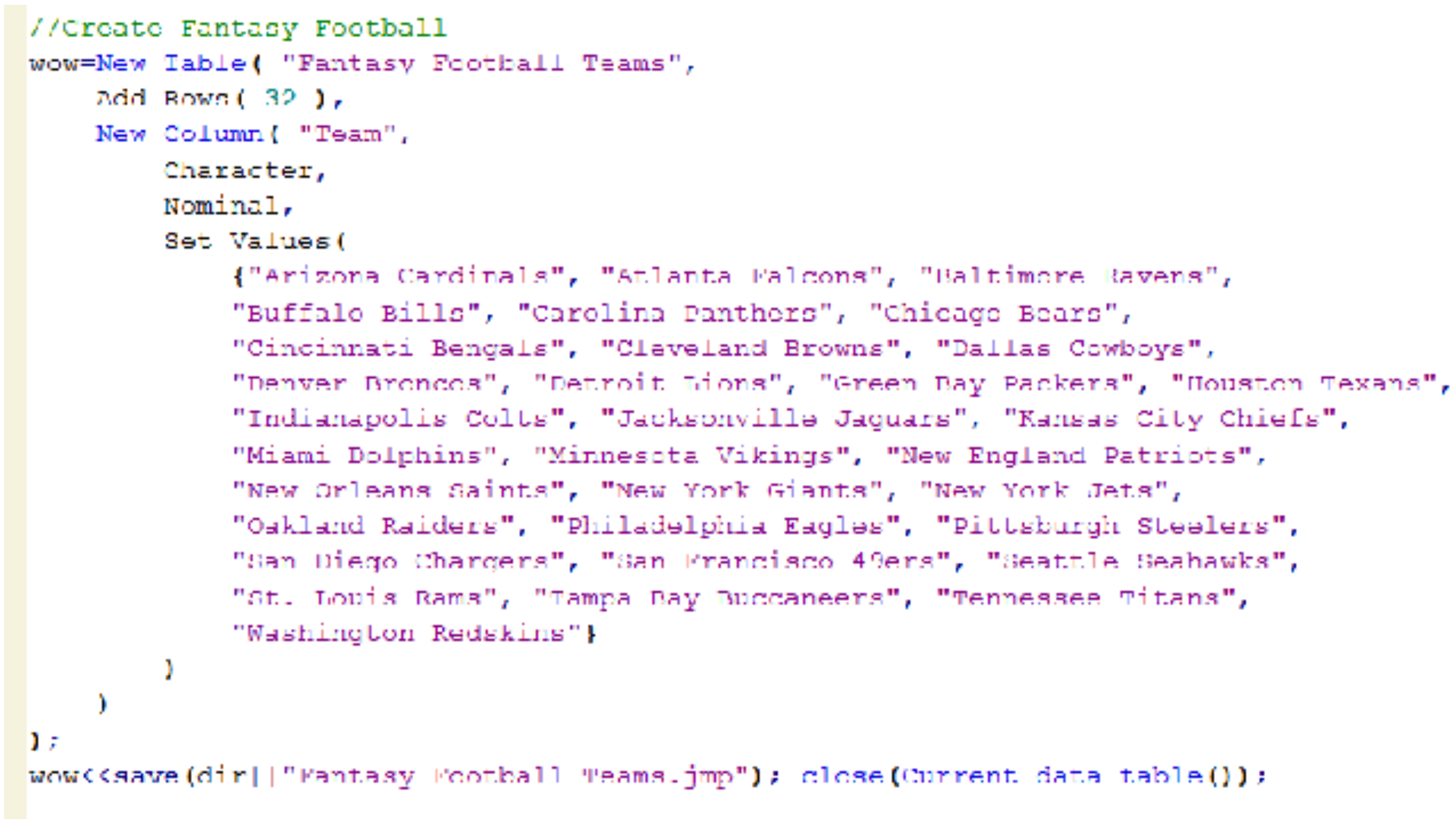


```
stn9=Data Table("AFCE") << Concatenate(
                Data Table("AFCW"),
                Data Table("AFCN"),
                Data Table("AFCS"),
                Data Table("NFCE"),
                Data Table("NFCW"),
                Data Table("NFCN"),
                Data Table("NFCS"),
                Output Table("Total Standings1")
);
stn9<<Sort(By(:Team),Order(Ascending),Output Table Name("Total Standings"));
```

Create a Fresh Table

- Add the appropriate number of rows: one for each team
- Create columns with various attributes
  - In this example, JMP simply creates a column of team names in a new table
  - It is often useful to create a new table to ensure the data are used appropriately during script execution, and are not accidentally lost or altered

The Code: Create a Fresh Table

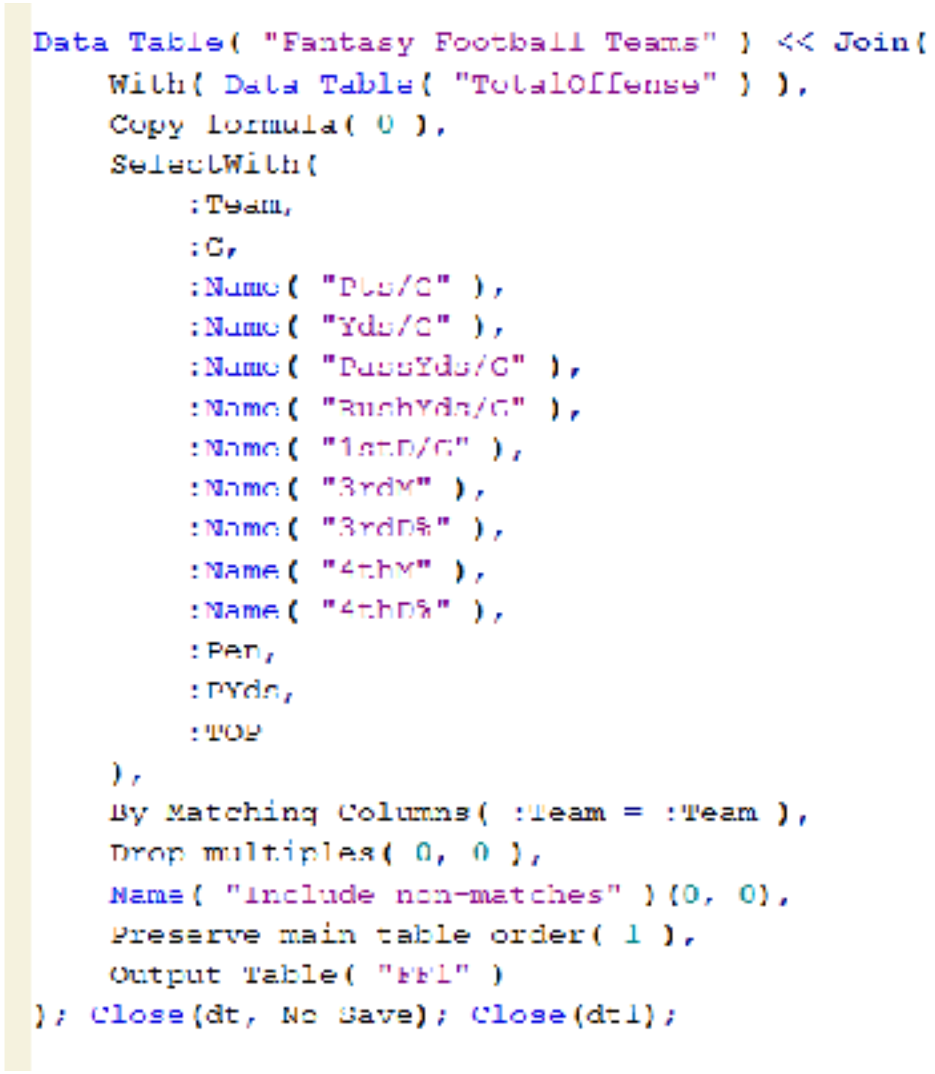


```
//Create Fantasy Football
wow=New Table( "Fantasy Football Teams",
    Add Rows( 32 ),
    New Column( "Team",
        Character,
        Nominal,
        Set Values(
            {"Arizona Cardinals", "Atlanta Falcons", "Baltimore Ravens",
            "Buffalo Bills", "Carolina Panthers", "Chicago Bears",
            "Cincinnati Bengals", "Cleveland Browns", "Dallas Cowboys",
            "Denver Broncos", "Detroit Lions", "Green Bay Packers", "Houston Texans",
            "Indianapolis Colts", "Jacksonville Jaguars", "Kansas City Chiefs",
            "Miami Dolphins", "Minnesota Vikings", "New England Patriots",
            "New Orleans Saints", "New York Giants", "New York Jets",
            "Oakland Raiders", "Philadelphia Eagles", "Pittsburgh Steelers",
            "San Diego Chargers", "San Francisco 49ers", "Seattle Seahawks",
            "St. Louis Rams", "Tampa Bay Buccaneers", "Tennessee Titans",
            "Washington Redskins"}
        )
    )
);
wow<<save(dir||"Fantasy Football Teams.jmp"); close(Current data table());
```

Joining Tables

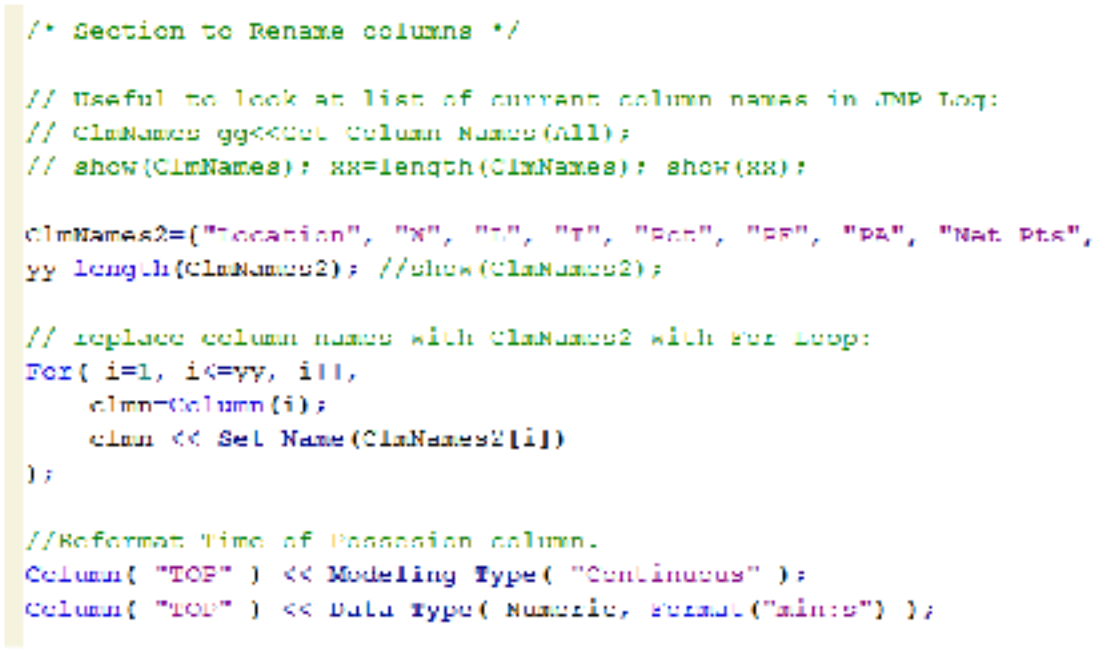
- This can be annoying without JMP:
  - JMP has a nice GUI for joining tables – Tables platform
- Naming conventions are important, but…
  - JMP automatically renames columns originating from different tables that share the same name
- Once you have joined the tables, simply copy and paste the table script of the resultant table into your larger, custom script
- Repeat, as necessary, for each join

The Code: Joining Tables



```
Data Table( "Fantasy Football Teams" ) << Join(
	With( Data Table( "TotalOffense" ) ),
	Copy formula( 0 ),
	SelectWith(
		:Team,
		:G,
		:Name( "Pts/G" ),
		:Name( "Yds/G" ),
		:Name( "PassYds/G" ),
		:Name( "RushYds/G" ),
		:Name( "1stD/G" ),
		:Name( "3rdM" ),
		:Name( "3rdD%" ),
		:Name( "4thM" ),
		:Name( "4thD%" ),
		:Pen,
		:PYds,
		:TOP
	),
	By Matching Columns( :Team = :Team ),
	Drop multiples( 0, 0 ),
	Name( "Include non-matches" )(0, 0),
	Preserve main table order( 1 ),
	Output Table( "FF1" )
); Close(dt, No Save); Close(dt1);
```

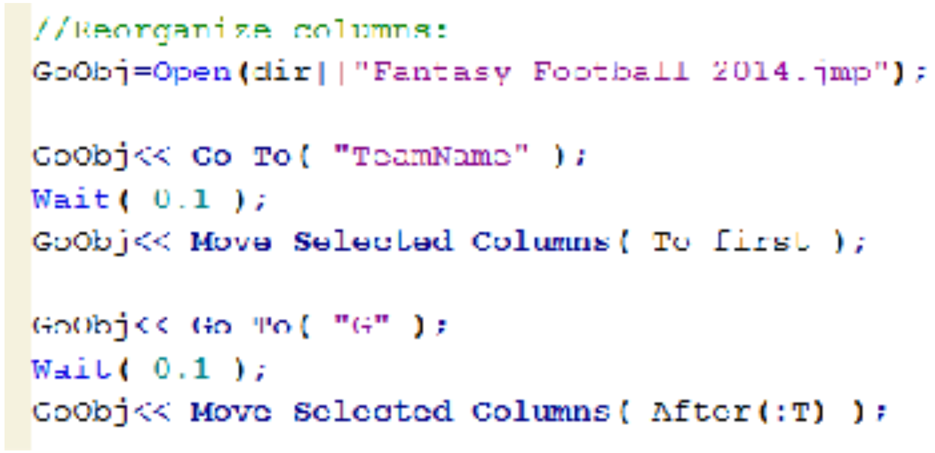
Rename & Reformat Columns



```
/* Section to Rename columns */

// Useful to look at list of current column names in JMP Log:
// ClmNames gg<<Get Column Names(All);
// show(ClmNames); xx=length(ClmNames); show(xx);

ClmNames2={"Location", "X", "Y", "T", "Pct", "PF", "PA", "Net Pts",
yy length(ClmNames2); //show(ClmNames2);

// replace column names with ClmNames2 with For Loop:
For( i=1, i<=yy, i++,
     clmn=Column(i);
     clmn << Set Name(ClmNames2[i])
);

//Reformat Time of Possession column.
Column( "TOP" ) << Modeling Type( "Continuous" );
Column( "TOP" ) << Data Type( Numeric, Format("min:s") );
```

- To replace odd-sounding column names:
- First, create a list of preferred names, then rename the columns using the combination of a "For Loop" and the "Set Name" function

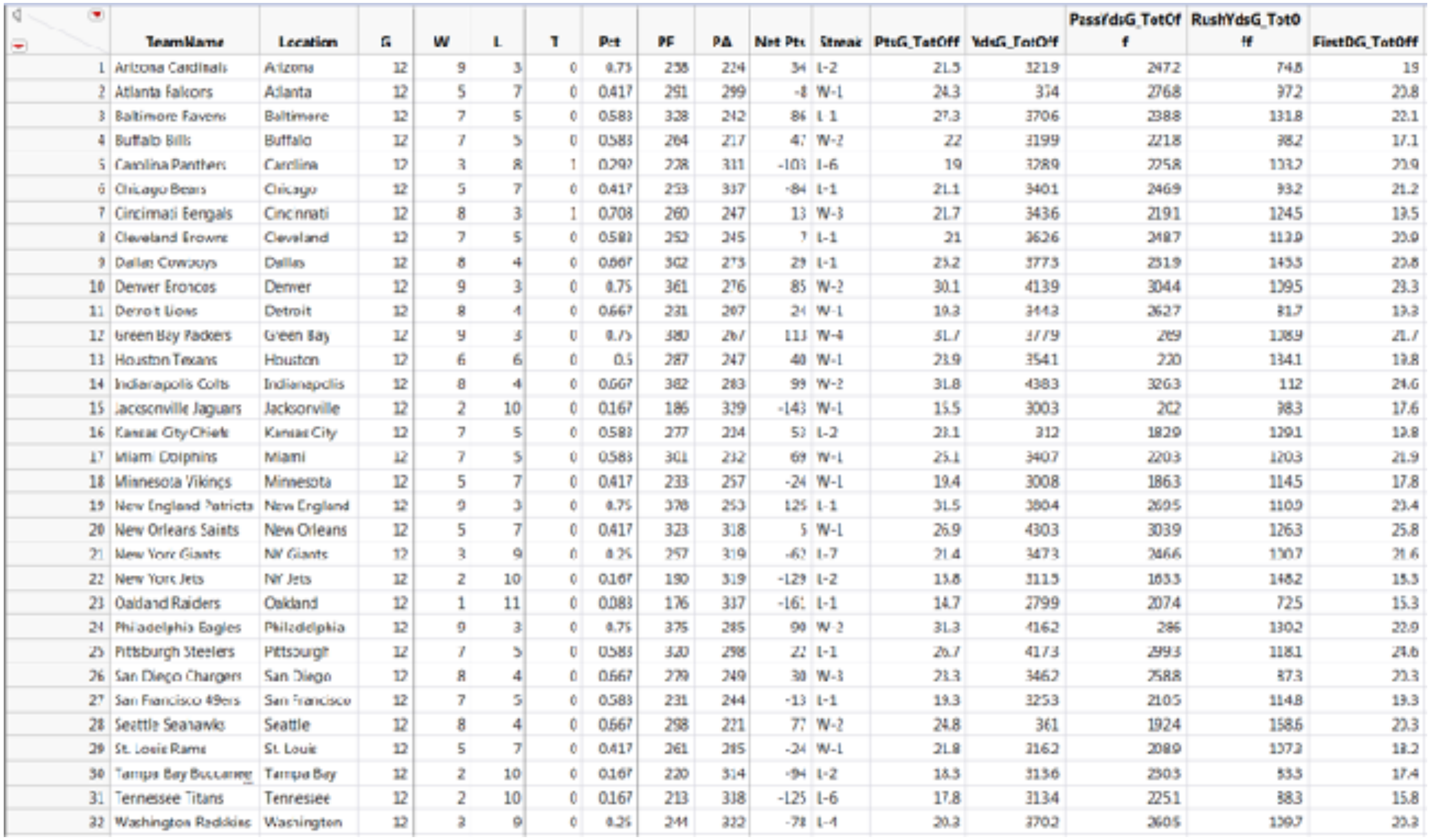
Automate Column Reorganization



```
//Reorganize columns:
GoObj=Open(dir||"Fantasy Football 2014.jmp");

GoObj<< Go To( "TeamName" );
Wait( 0.1 );
GoObj<< Move Selected Columns( To first );

GoObj<< Go To( "G" );
Wait( 0.1 );
GoObj<< Move Selected Columns( After(:T) );
```

- Once the data is collected and new columns are created, one may want to change the order of columns in the data table
- It is nice to do this automatically with JSL
- Easy with “Move Selected Columns” command

Data Table View

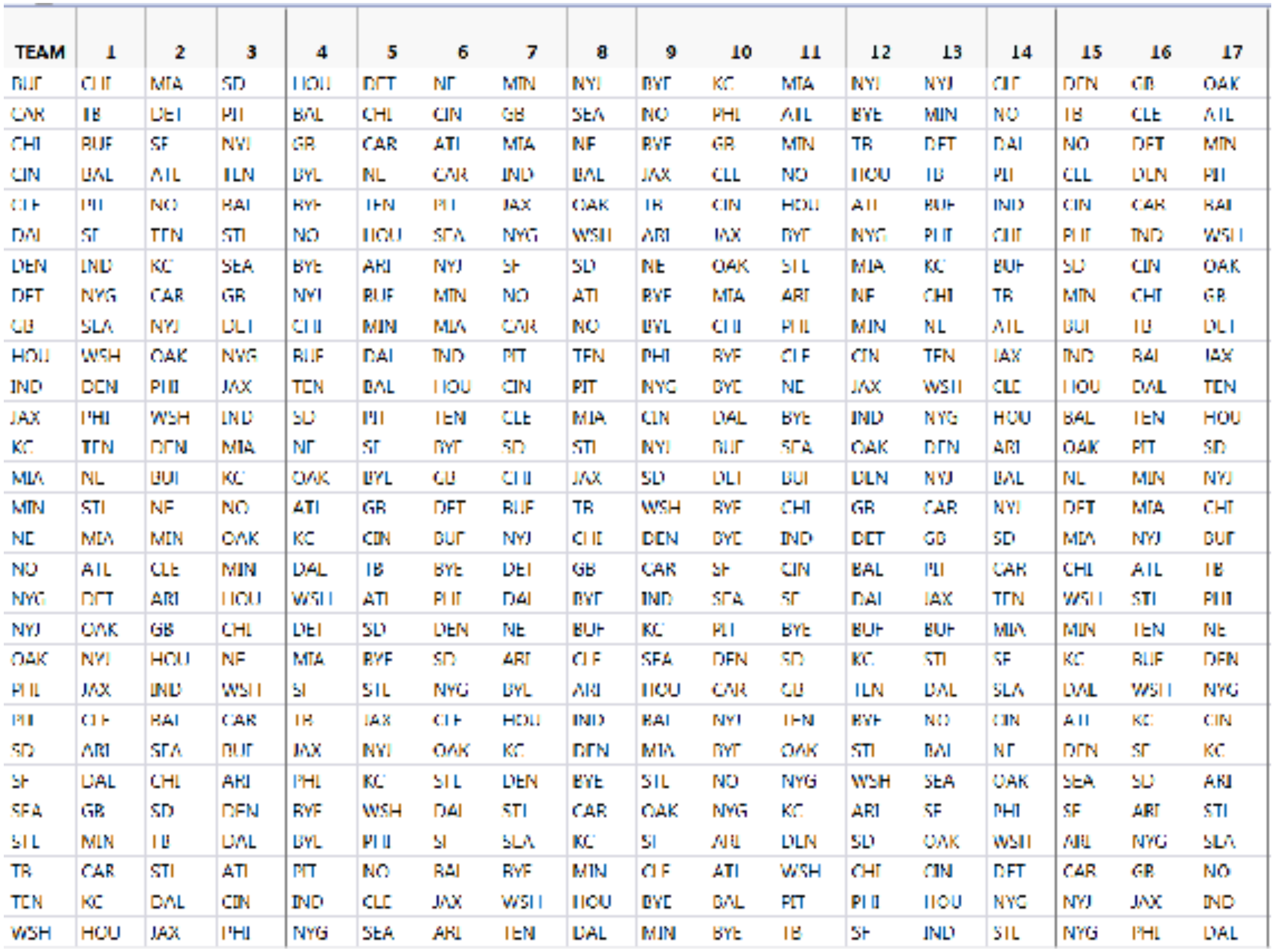


	TeamName	Location	G	W	L	T	Pct	PF	PA	Net Pts	Streak	PtsG_TotOff	YdsG_TotOff	PassYdsG_TotOff	RushYdsG_TotOff	FirstDG_TotOff
1	Arizona Cardinals	Arizona	12	9	3	0	0.75	258	224	34	L-2	21.5	321.9	247.2	74.8	19
2	Atlanta Falcons	Atlanta	12	5	7	0	0.417	291	299	-8	W-1	24.3	374	276.8	97.2	20.8
3	Baltimore Ravens	Baltimore	12	7	5	0	0.583	328	242	86	L-1	27.3	370.6	238.8	131.8	22.1
4	Buffalo Bills	Buffalo	12	7	5	0	0.583	264	217	47	W-2	22	319.9	221.8	98.2	17.1
5	Carolina Panthers	Carolina	12	3	8	1	0.292	228	311	-103	L-6	19	328.9	225.8	133.2	20.9
6	Chicago Bears	Chicago	12	5	7	0	0.417	253	337	-84	L-1	21.1	340.1	246.9	93.2	21.2
7	Cincinnati Bengals	Cincinnati	12	8	3	1	0.708	260	247	13	W-3	21.7	343.6	219.1	124.5	19.5
8	Cleveland Browns	Cleveland	12	7	5	0	0.583	252	245	7	L-1	21	362.6	248.7	113.9	20.9
9	Dallas Cowboys	Dallas	12	8	4	0	0.667	302	273	29	L-1	25.2	377.3	231.9	145.5	20.8
10	Denver Broncos	Denver	12	9	3	0	0.75	361	276	85	W-2	30.1	413.9	304.4	109.5	23.3
11	Detroit Lions	Detroit	12	8	4	0	0.667	231	207	24	W-1	19.3	344.3	262.7	81.7	19.3
12	Green Bay Packers	Green Bay	12	9	3	0	0.75	380	267	113	W-4	31.7	377.9	269	108.9	21.7
13	Houston Texans	Houston	12	6	6	0	0.5	287	247	40	W-1	23.9	354.1	220	134.1	19.8
14	Indianapolis Colts	Indianapolis	12	8	4	0	0.667	382	283	99	W-2	31.8	438.3	326.3	112	24.6
15	Jacksonville Jaguars	Jacksonville	12	2	10	0	0.167	186	329	-143	W-1	15.5	300.3	202	98.3	17.6
16	Kansas City Chiefs	Kansas City	12	7	5	0	0.583	277	224	53	L-2	23.1	312	182.9	129.1	19.8
17	Miami Dolphins	Miami	12	7	5	0	0.583	301	232	69	W-1	25.1	340.7	220.3	120.3	21.9
18	Minnesota Vikings	Minnesota	12	5	7	0	0.417	233	257	-24	W-1	19.4	300.8	186.3	114.5	17.8
19	New England Patriots	New England	12	9	3	0	0.75	378	253	125	L-1	31.5	380.4	269.5	110.9	23.4
20	New Orleans Saints	New Orleans	12	5	7	0	0.417	323	318	5	W-1	26.9	430.3	303.9	126.3	25.8
21	New York Giants	NY Giants	12	3	9	0	0.25	257	319	-62	L-7	21.4	347.3	246.6	100.7	21.6
22	New York Jets	NY Jets	12	2	10	0	0.167	190	319	-129	L-2	15.8	311.5	163.3	148.2	15.3
23	Oakland Raiders	Oakland	12	1	11	0	0.083	176	337	-161	L-1	14.7	279.9	207.4	72.5	15.3
24	Philadelphia Eagles	Philadelphia	12	9	3	0	0.75	375	285	90	W-2	31.3	416.2	286	130.2	22.9
25	Pittsburgh Steelers	Pittsburgh	12	7	5	0	0.583	320	298	22	L-1	26.7	417.3	299.3	118.1	24.6
26	San Diego Chargers	San Diego	12	8	4	0	0.667	279	249	30	W-3	23.3	346.2	258.8	87.3	20.3
27	San Francisco 49ers	San Francisco	12	7	5	0	0.583	231	244	-13	L-1	19.3	325.3	210.5	114.8	19.3
28	Seattle Seahawks	Seattle	12	8	4	0	0.667	298	221	77	W-2	24.8	361	192.4	158.6	20.3
29	St. Louis Rams	St. Louis	12	5	7	0	0.417	261	285	-24	W-1	21.8	316.2	208.9	107.3	18.2
30	Tampa Bay Buccanee	Tampa Bay	12	2	10	0	0.167	220	314	-94	L-2	18.3	313.6	230.3	83.3	17.4
31	Tennessee Titans	Tennessee	12	2	10	0	0.167	213	338	-125	L-6	17.8	313.4	225.1	88.3	15.8
32	Washington Redskins	Washington	12	3	9	0	0.25	244	322	-78	L-4	20.3	370.2	260.5	109.7	20.3

Develop the Associative Array

- Rationale: I wanted to compare information contained in cells located in different rows and columns of my table
- Associative array was very useful for this task
- In this case, the abbreviated city name for each team is a key
  - Values found in another column called “Strength”, in the same row.
- My goal was to predict how strong a team’s future performance would be, not only based on their own past performance,
- But, also relative to their future opponents’ past performance
- “Strength” is calculated from a combination of wins & net pts

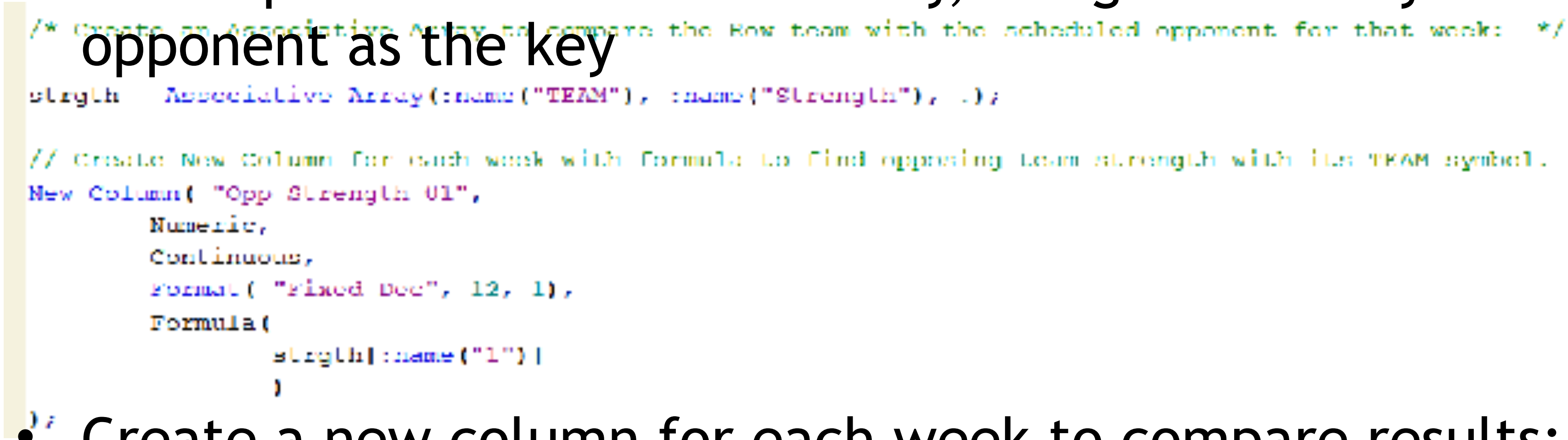
17-week schedule: Keys



TEAM	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
BUF	CHI	MIA	SD	HOU	DET	NE	MIN	NYJ	BYE	KC	MIA	NYJ	NYJ	CLE	DEN	GB	OAK
CAR	TB	DET	PIT	BAL	CHI	CIN	GB	SEA	NO	PHI	ATL	BYE	MIN	NO	TB	CLE	ATL
CHI	BUF	SF	NYJ	GB	CAR	ATL	MIA	NE	BYE	GB	MIN	TB	DET	DAL	NO	DET	MIN
CIN	BAL	ATL	TEN	BYE	NE	CAR	IND	BAL	JAX	CLE	NO	HOU	TB	PIT	CLE	DEN	PIT
CLE	PIT	NO	BAL	BYE	TEN	PIT	JAX	OAK	TB	CIN	HOU	ATL	BUF	IND	CIN	CAR	BAL
DAL	SF	TEN	STL	NO	HOU	SEA	NYG	WSH	ARI	JAX	BYE	NYG	PHI	CHI	PHI	IND	WSH
DEN	IND	KC	SEA	BYE	ARI	NYJ	SF	SD	NE	OAK	STL	MIA	KC	BUF	SD	CIN	OAK
DET	NYG	CAR	GB	NYJ	BUF	MIN	NO	ATL	BYE	MIA	ARI	NE	CHI	TB	MIN	CHI	GB
GB	SEA	NYJ	DET	CHI	MIN	MIA	CAR	NO	BYE	CHI	PHI	MIN	NE	ATL	BUF	TB	DET
HOU	WSH	OAK	NYG	BUF	DAL	IND	PIT	TEN	PHI	BYE	CLE	CIN	TEN	JAX	IND	BAL	JAX
IND	DEN	PHI	JAX	TEN	BAL	HOU	CIN	PIT	NYG	BYE	NE	JAX	WSH	CLE	HOU	DAL	TEN
JAX	PHI	WSH	IND	SD	PIT	TEN	CLE	MIA	CIN	DAL	BYE	IND	NYG	HOU	BAL	TEN	HOU
KC	TEN	DEN	MIA	NE	SF	BYE	SD	STL	NYJ	BUF	SEA	OAK	DEN	ARI	OAK	PIT	SD
MIA	NE	BUF	KC	OAK	BYE	GB	CHI	JAX	SD	DET	BUF	DEN	NYJ	BAL	NE	MIN	NYJ
MIN	STL	NE	NO	ATL	GB	DET	BUF	TB	WSH	BYE	CHI	GB	CAR	NYJ	DET	MIA	CHI
NE	MIA	MIN	OAK	KC	CIN	BUF	NYJ	CHI	DEN	BYE	IND	DET	GB	SD	MIA	NYJ	BUF
NO	ATL	CLE	MIN	DAL	TB	BYE	DET	GB	CAR	SF	CIN	BAL	PIT	CAR	CHI	ATL	TB
NYG	DET	ARI	HOU	WSH	ATL	PHI	DAL	BYE	IND	SEA	SF	DAL	JAX	TEN	WSH	STL	PHI
NYJ	OAK	GB	CHI	DET	SD	DEN	NE	BUF	KC	PIT	BYE	BUF	BUF	MIA	MIN	TEN	NE
OAK	NYJ	HOU	NE	MIA	BYE	SD	ARI	CLE	SEA	DEN	SD	KC	STL	SF	KC	BUF	DEN
PHI	JAX	IND	WSH	SF	STL	NYG	BYE	ARI	HOU	CAR	GB	TEN	DAL	SEA	DAL	WSH	NYG
PIT	CLE	BAL	CAR	TB	JAX	CLE	HOU	IND	BAL	NYJ	TEN	BYE	NO	CIN	ATL	KC	CIN
SD	ARI	SEA	BUF	JAX	NYJ	OAK	KC	DEN	MIA	BYE	OAK	STL	BAL	NE	DEN	SF	KC
SF	DAL	CHI	ARI	PHI	KC	STL	DEN	BYE	STL	NO	NYG	WSH	SEA	OAK	SEA	SD	ARI
SEA	GB	SD	DEN	BYE	WSH	DAL	STL	CAR	OAK	NYG	KC	ARI	SF	PHI	SF	ARI	STL
STL	MIN	TB	DAL	BYE	PHI	SF	SEA	KC	SF	ATL	DEN	SD	OAK	WSH	ATL	NYG	SEA
TB	CAR	STL	ATL	PIT	NO	BAL	BYE	MIN	CLE	ATL	WSH	CHI	CIN	DET	CAR	GB	NO
TEN	KC	DAL	CIN	IND	CLE	JAX	WSH	HOU	BYE	BAL	PIT	PHI	HOU	NYG	NYJ	JAX	IND
WSH	HOU	JAX	PHI	NYG	SEA	ARI	TEN	DAL	MIN	BYE	TB	SF	IND	STL	NYG	PHI	DAL

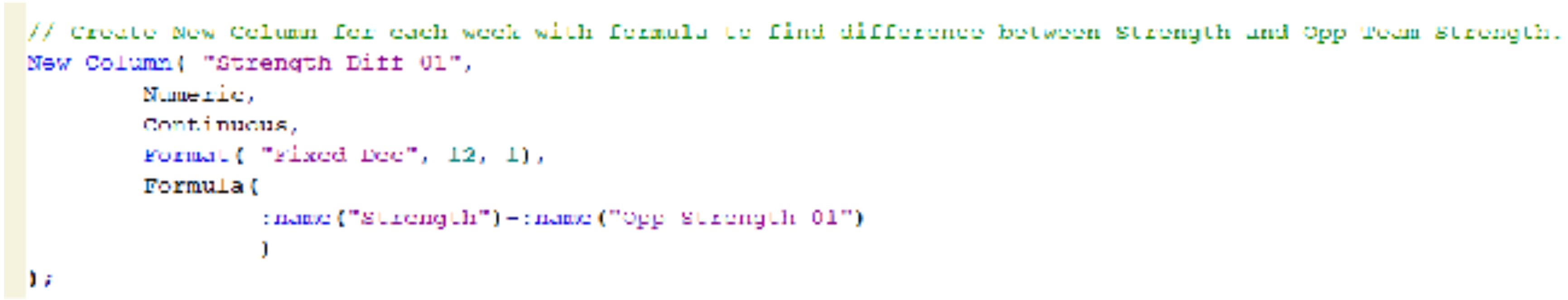
The Code: Simple Associative Array

- Create the associative array, then a formula column
- The formula references the correct value through a subscript of the associative array, using the weekly opponent as the key



```
/* Create an Associative Array to compare the Row team with the scheduled opponent for that week:  */

strgth   Associative Array(:name("TEAM"), :name("Strength"), .);

// Create New Column for each week with formula to find opposing team strength with its TEAM symbol.
New Column( "Opp Strength 01",
        Numeric,
        Continuous,
        Format( "Fixed Dec", 12, 1),
        Formula(
                strgth[:name("1")]
                )
);
```

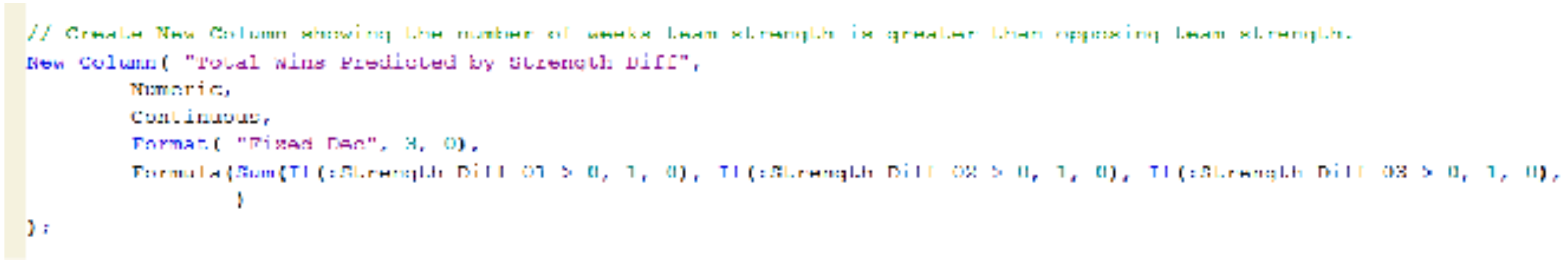
- Create a new column for each week to compare results:



```
// Create New Column for each week with formula to find difference between Strength and Opp Team Strength.
New Column( "Strength Diff 01",
        Numeric,
        Continuous,
        Format( "Fixed Dec", 12, 1),
        Formula(
                :name("Strength")-:name("Opp Strength 01")
                )
);
```

Counting Predicted Wins, by Column

- A simple way to count column results:
  - Create a new formula column
  - Sum a series of If-then conditions for each column value



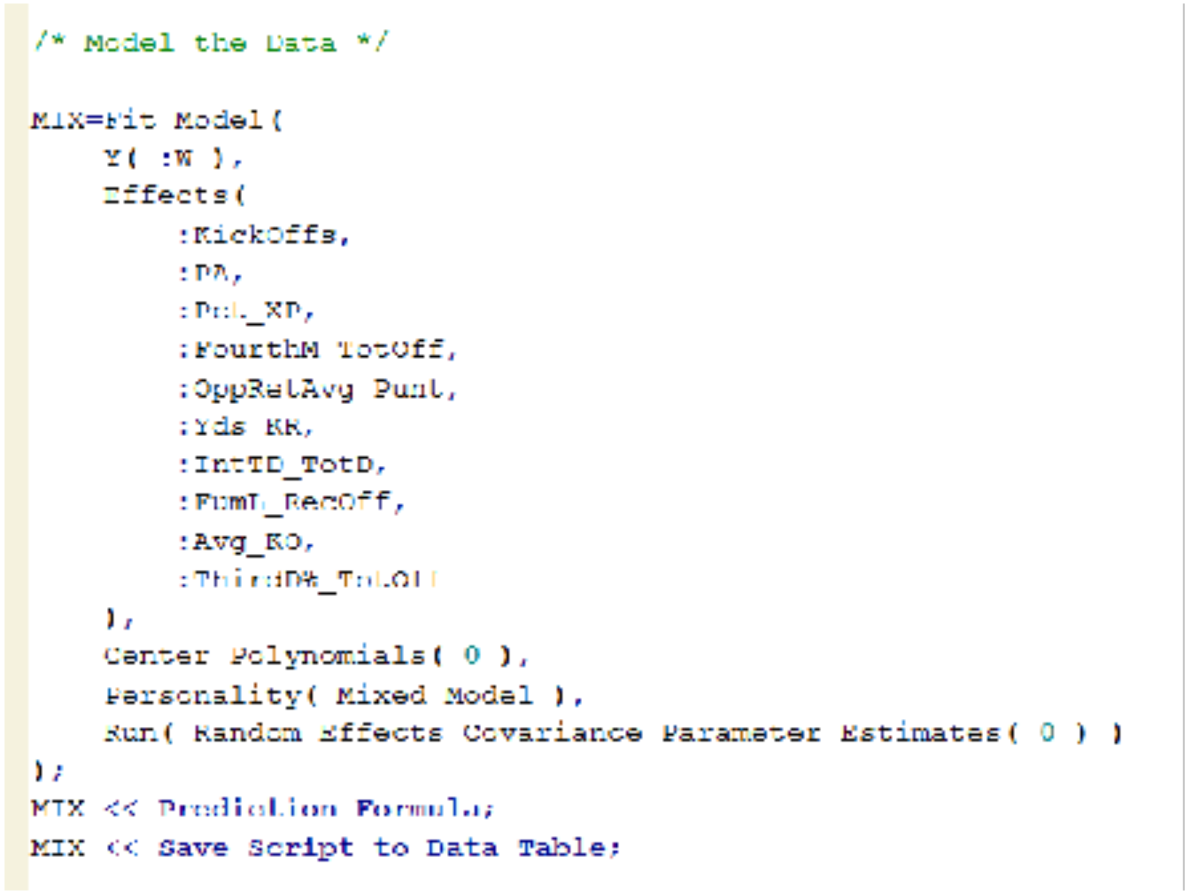
```
// Create New Column showing the number of weeks team strength is greater than opposing team strength.
New Column( "Total Wins Predicted by Strength Diff",
        Numeric,
        Continuous,
        Format( "Fixed Dec", 3, 0),
        Formula(Sum(If(:Strength Diff 01 > 0, 1, 0), If(:Strength Diff 02 > 0, 1, 0), If(:Strength Diff 03 > 0, 1, 0),
                )
);
```

In this case, if the difference in Strength was positive,
 the prediction was considered a win and counted.

Create Predictive Models

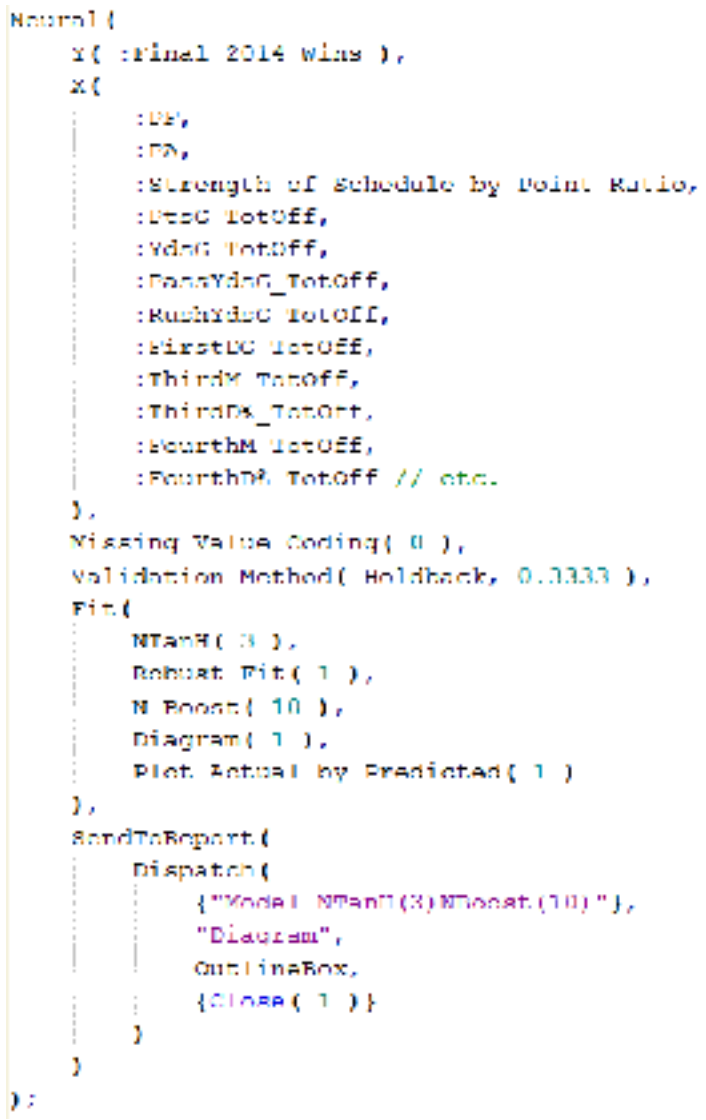
- Use any of several JMP modeling techniques to create predictions:
  - Continuous: Fit Model platform
  - Classification: Neural Net, etc.
- Develop the optimal model in one of JMP's platforms,
- Save the script to the data table, or a new script window, and
- Copy and paste the model script into custom script.
- One can set the model to run automatically, or
- Save future model scripts to the data table, with JSL–
- The model will be available in the list of data table scripts of the newly created table.

Create Model & Save to Table



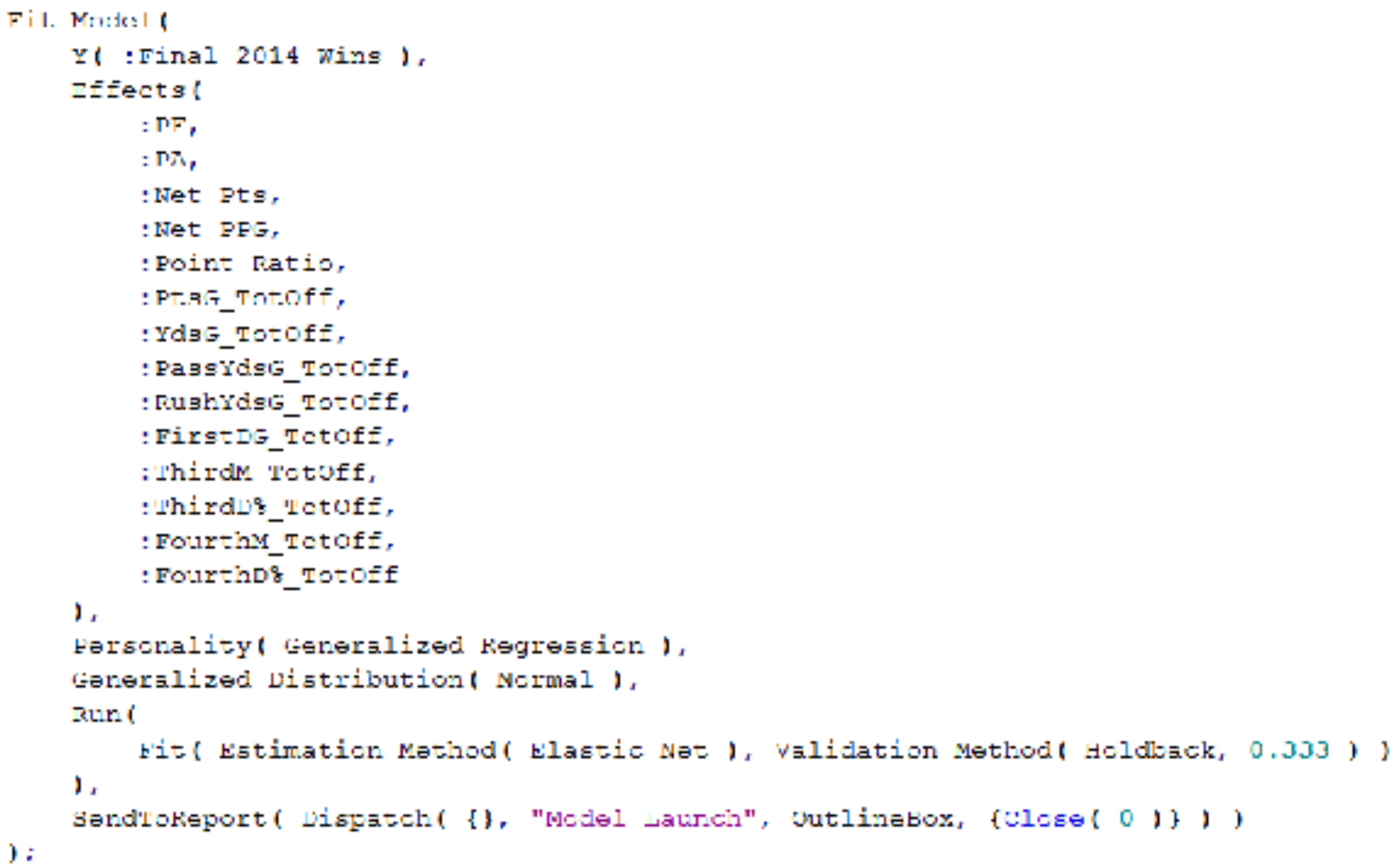
```
/* Model the Data */

MIX=Fit Model(
     Y( :W ),
     Effects(
          :KickOffs,
          :PA,
          :Pct_XP,
          :FourthM TotOff,
          :OppRelAvg Punt,
          :Yds KR,
          :IntTD_TotD,
          :FumL_RecOff,
          :Avg_KO,
          :ThirdD%_TotOff
     ),
     Center Polynomials( 0 ),
     Personality( Mixed Model ),
     Run( Random Effects Covariance Parameter Estimates( 0 ) )
);
MIX << Prediction Formula;
MIX << Save Script to Data Table;
```

Boosted Neural Net – JMP Pro



```
Neural(
	Y( :Final 2014 Wins ),
	X(
		:DF,
		:PA,
		:Strength of Schedule by Point Ratio,
		:PtsG TotOff,
		:YdsG TotOff,
		:PassYdsG_TotOff,
		:RushYdsG TotOff,
		:FirstDG TotOff,
		:ThirdM TotOff,
		:ThirdPk_TotOff,
		:FourthM TotOff,
		:FourthPk TotOff // etc.
	),
	Missing Value Coding( 0 ),
	Validation Method( Holdback, 0.3333 ),
	Fit(
		NTanH( 3 ),
		Robust Fit( 1 ),
		N Boost( 10 ),
		Diagram( 1 ),
		Plot Actual by Predicted( 1 )
	),
	SendToReport(
		Dispatch(
			{"Model NTanH(3)NBoost(10)"},
			"Diagram",
			OutlineBox,
			{Close( 1 )}
		)
	)
);
```

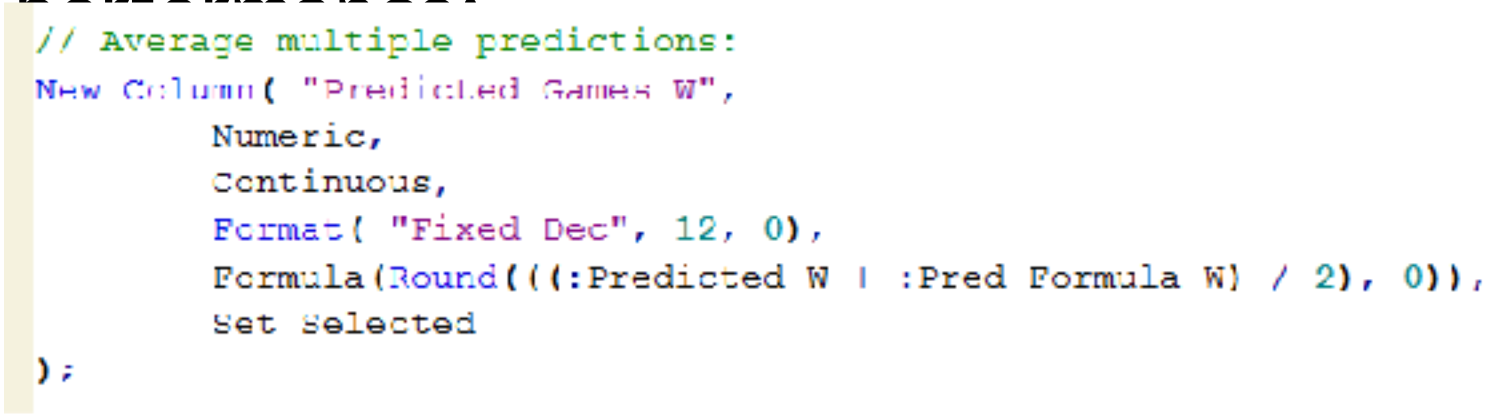
Elastic Net – JMP Pro



```
Fit Model(
	Y( :Final 2014 Wins ),
	Effects(
		:PF,
		:PA,
		:Net Pts,
		:Net PPG,
		:Point Ratio,
		:PtsG_TotOff,
		:YdsG_TotOff,
		:PassYdsG_TotOff,
		:RushYdsG_TotOff,
		:FirstDG_TotOff,
		:ThirdM_TotOff,
		:ThirdD%_TotOff,
		:FourthM_TotOff,
		:FourthD%_TotOff
	),
	Personality( Generalized Regression ),
	Generalized Distribution( Normal ),
	Run(
		Fit( Estimation Method( Elastic Net ), Validation Method( Holdback, 0.333 ) )
	),
	SendToReport( Dispatch( {}, "Model Launch", OutlineBox, {Close( 0 )} ) )
);
```

Model Averaging

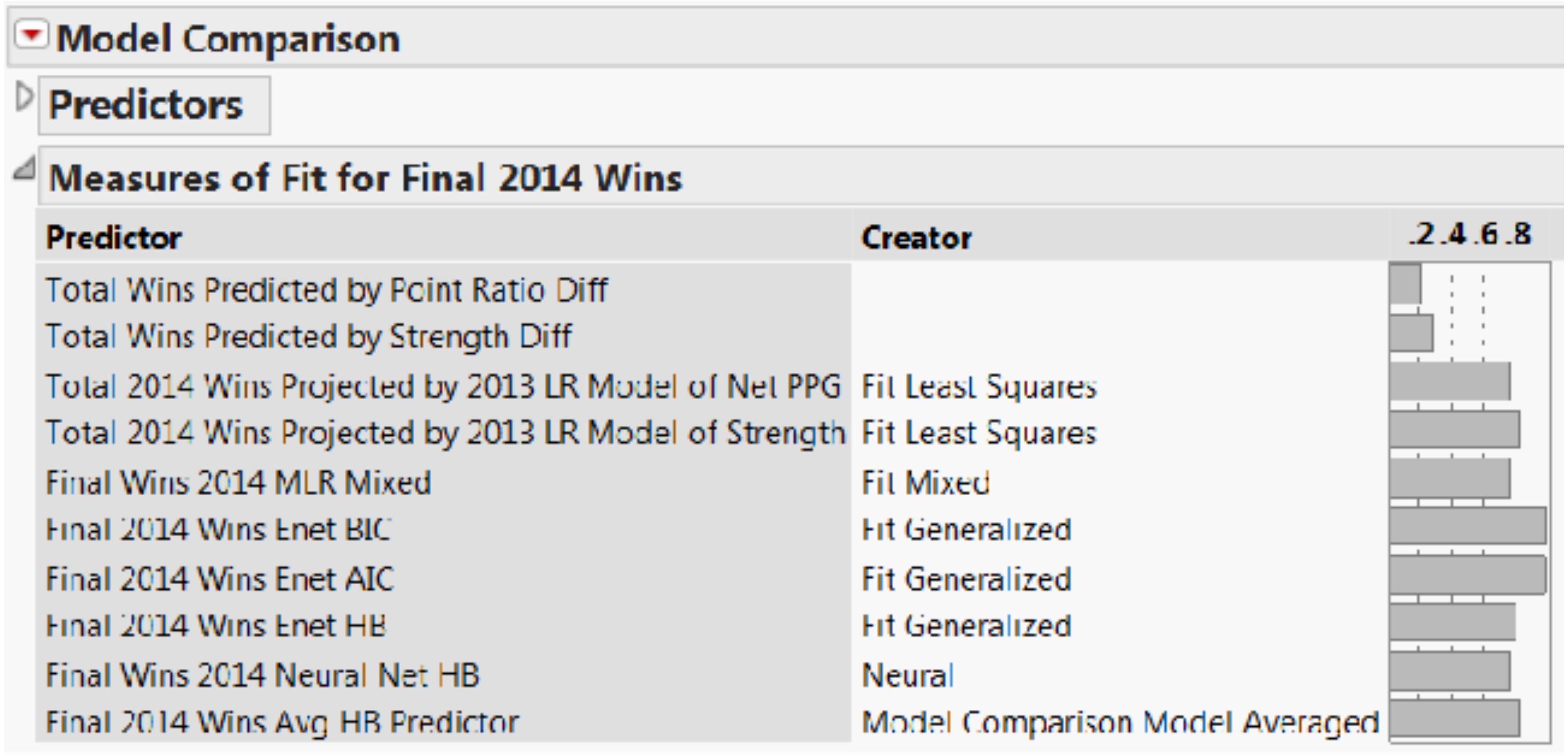
- The average of multiple model predictions may outperform individual model performance:



```
// Average multiple predictions:
New Column( "Predicted Games W",
        Numeric,
        Continuous,
        Format( "Fixed Dec", 12, 0),
        Formula(Round(((:Predicted W + :Pred Formula W) / 2), 0)),
        Set Selected
);
```

- JMP Pro performs this operation with a couple of clicks

Model Comparison – JMP Pro



Model Comparison

Predictors

Measures of Fit for Final 2014 Wins

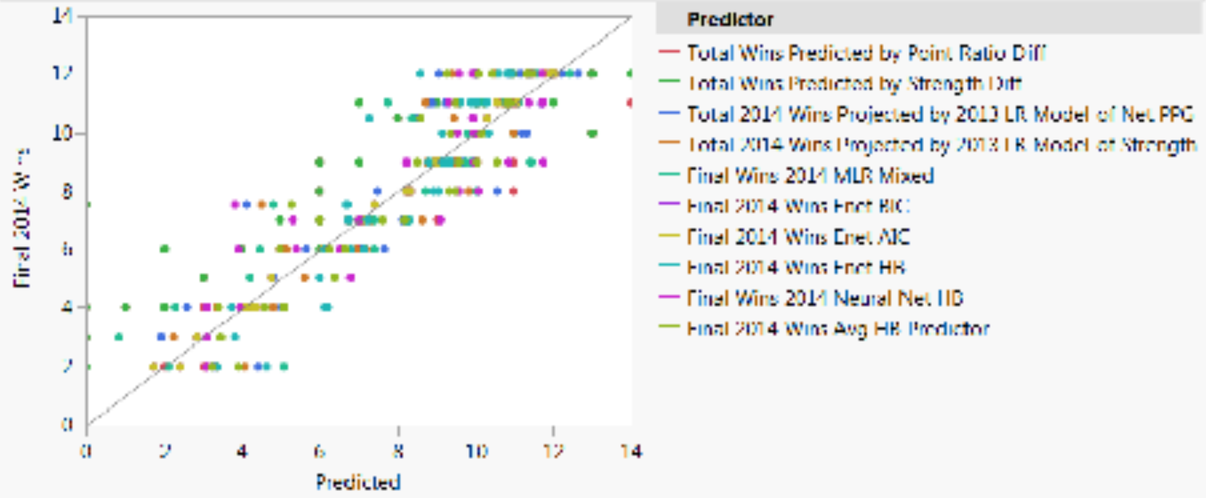
Predictor	Creator	.2 .4 .6 .8
Total Wins Predicted by Point Ratio Diff		
Total Wins Predicted by Strength Diff		
Total 2014 Wins Projected by 2013 LR Model of Net PPG	Fit Least Squares	
Total 2014 Wins Projected by 2013 LR Model of Strength	Fit Least Squares	
Final Wins 2014 MLR Mixed	Fit Mixed	
Final 2014 Wins Enet BIC	Fit Generalized	
Final 2014 Wins Enet AIC	Fit Generalized	
Final 2014 Wins Enet HB	Fit Generalized	
Final Wins 2014 Neural Net HB	Neural	
Final 2014 Wins Avg HB Predictor	Model Comparison Model Averaged	

Average of Neural Net HB and Elastic Net HB models performed slightly better

Model Comparison Graphics

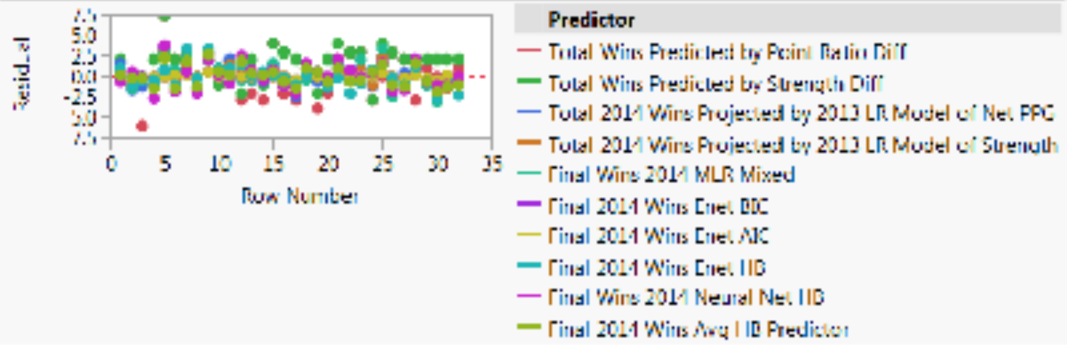
Actual by Predicted Plot

Actual by Predicted Plot for Final 2014 Wins



Residual by Row Plot

Residual by Row Plot for Final 2014 Wins



Conclusions

- With JSL, JMP can scrape data from multiple web sources, process the data, perform multiple predictions, compare them, and display them– all within a single script.
- The effective use of JSL can save a lot of time, especially when one must repeat an analysis on a regular basis.

Thank you!

- Thanks to Wendy Murphrey, for recommending the JSL reference sources
- Thanks to Peter Mroz, for showing me the usefulness of Associative Arrays
- Thank you for listening!
- Have a great JMP Discovery Summit!

Questions?