



A Reintroduction to Spline Modeling for Non-Linear Trends

Derek Montrichard

Canadian Imperial Bank of Commerce

May 2009

Introduction

The number one tool* of statistical modelers is regression, be it GLM, Logistic, etc.

Each form of regression attempts to solve for B to find the best fit for $Y = XB$

In each case, B is a linear transformation of X

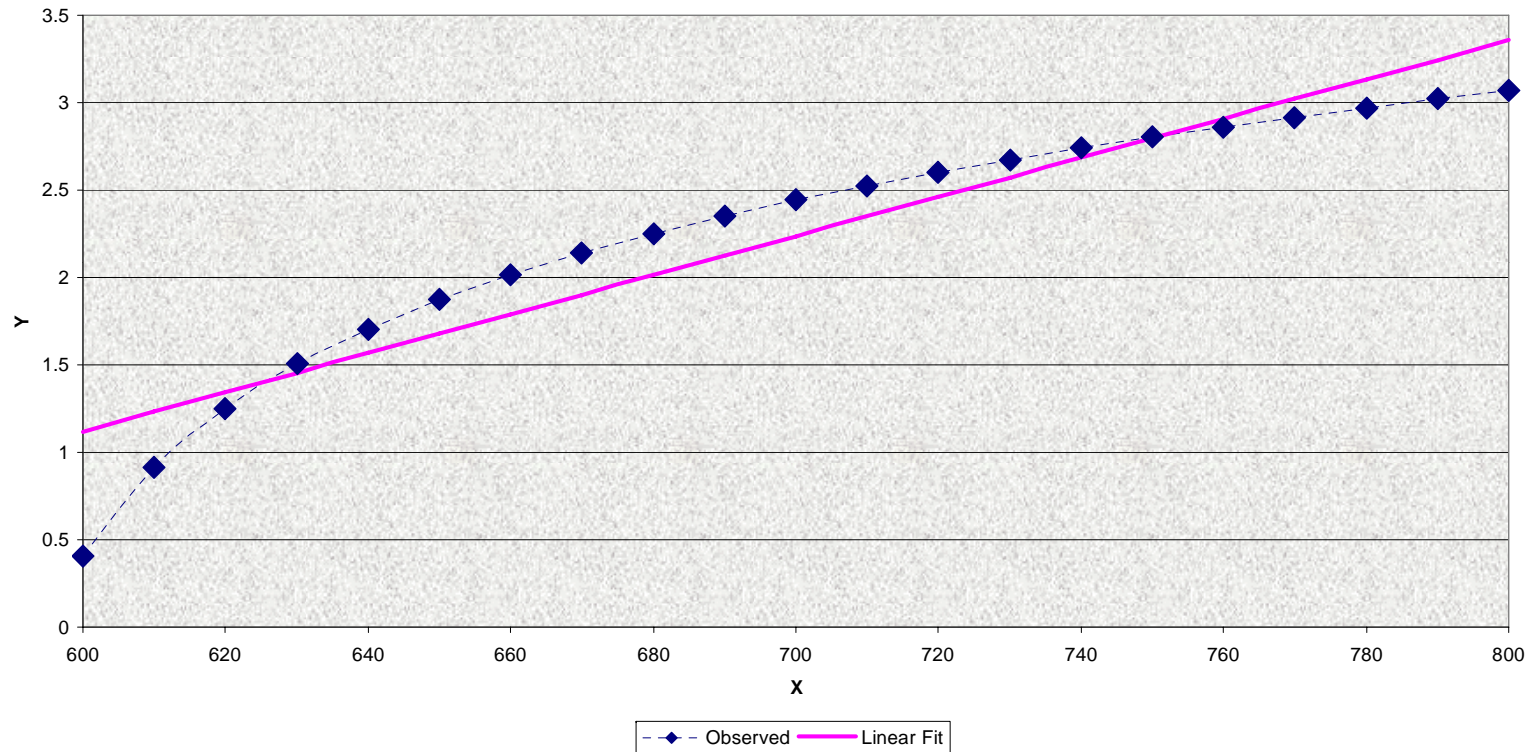
Y may not have a truly linear relationship with X

*as opposed to the number one tool of Starfleet



Example 1 – Non Linear Data

Spline Example 1 - Raw Data

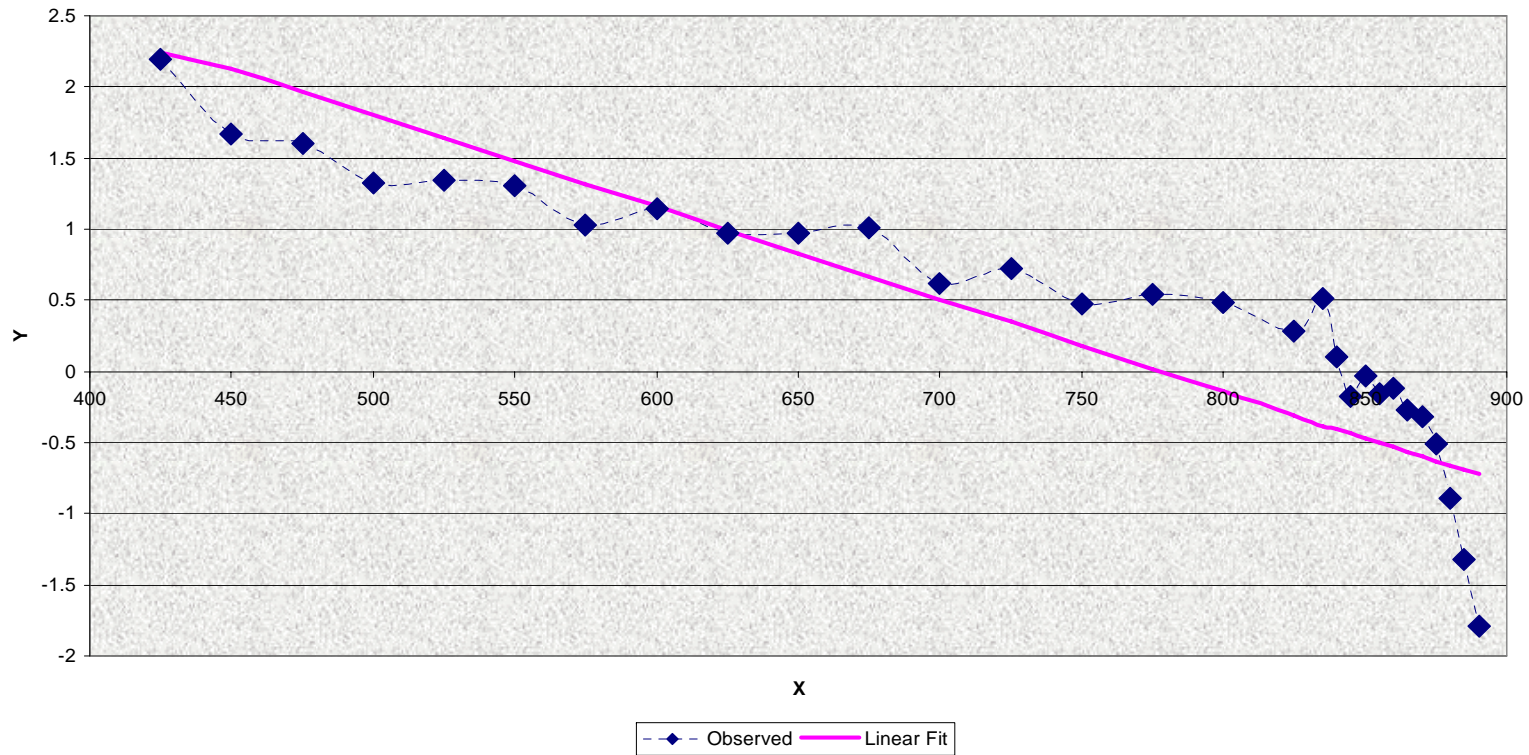


Linear model under predicts in the midrange, and over predicts in the extremes



Example 2 – Non Linear Data

Spline Example 2 - Raw Data



Bias clear in model... but how do we correct?



Smoothing approaches

Different methods attempt to help statisticians find and solve these non-linear trends (e.g. Box-Tidwell)

More frequent method: the analyst estimates relationships and create transformed variables (x_1^2 , $\log(x_1)$, $\sqrt{x_1}$, $1/x_1$, $x_1^3 \cdot \ln x_1$, $\sin(x_1)$, etc.) through trial and error

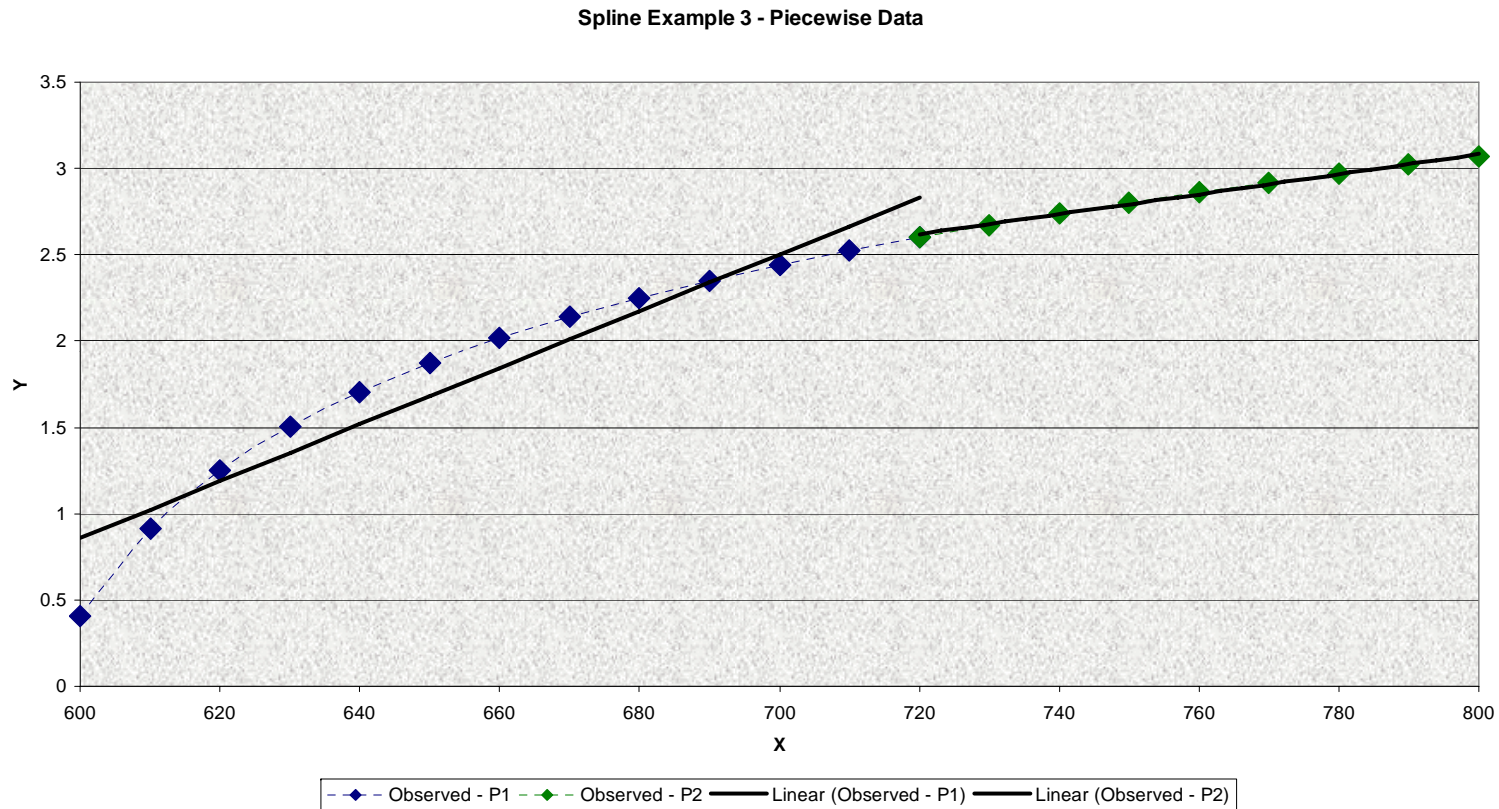


Q: What are splines?

- Splines create piecewise segments over X and fit linear trends on the individual components
- Key of splines : The relationship of Y vs. X remains continuous



Example 3 – Piecewise Lines

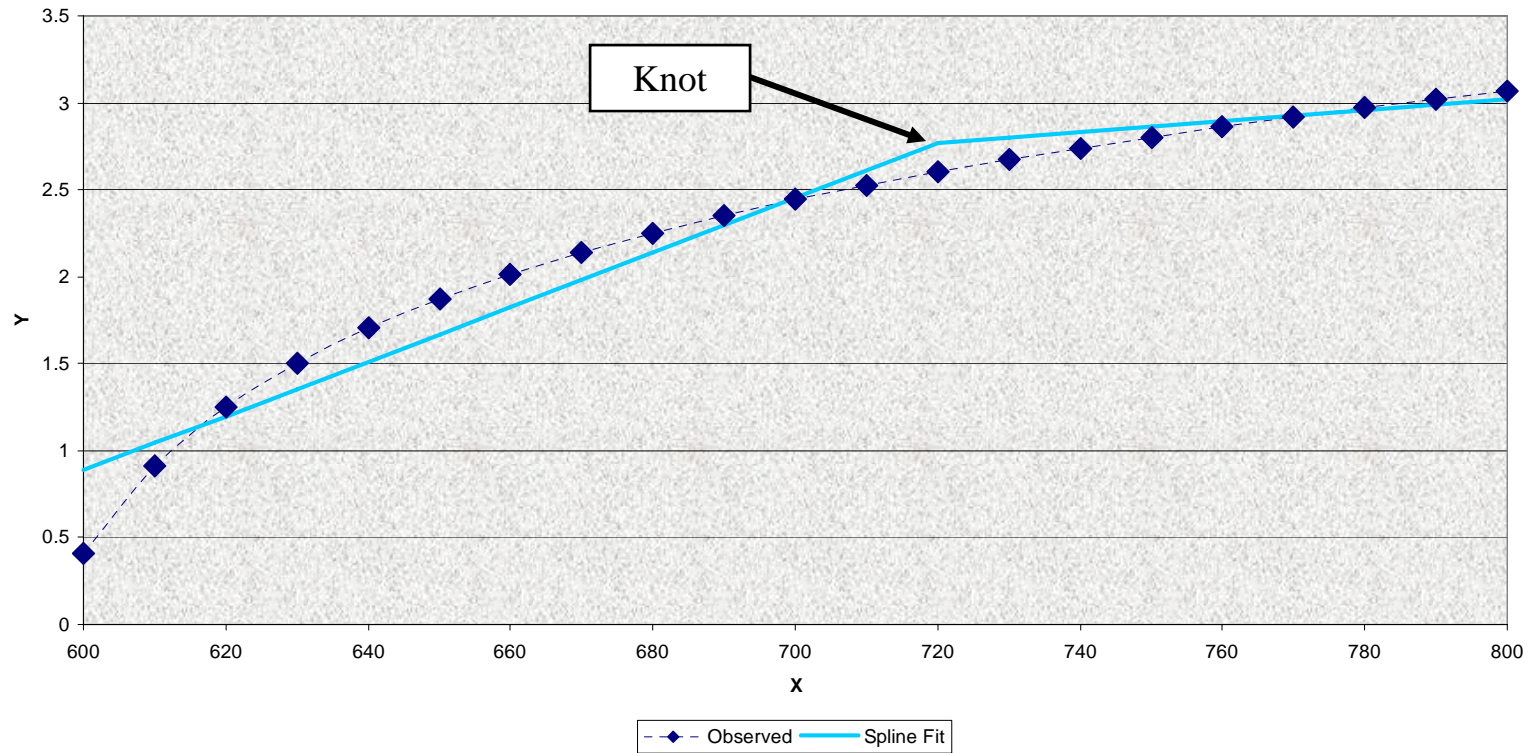


Model appears better, but not continuous at 720



Example 4 – Spline Data

Spline Example 4 - Two Piece Spline Data

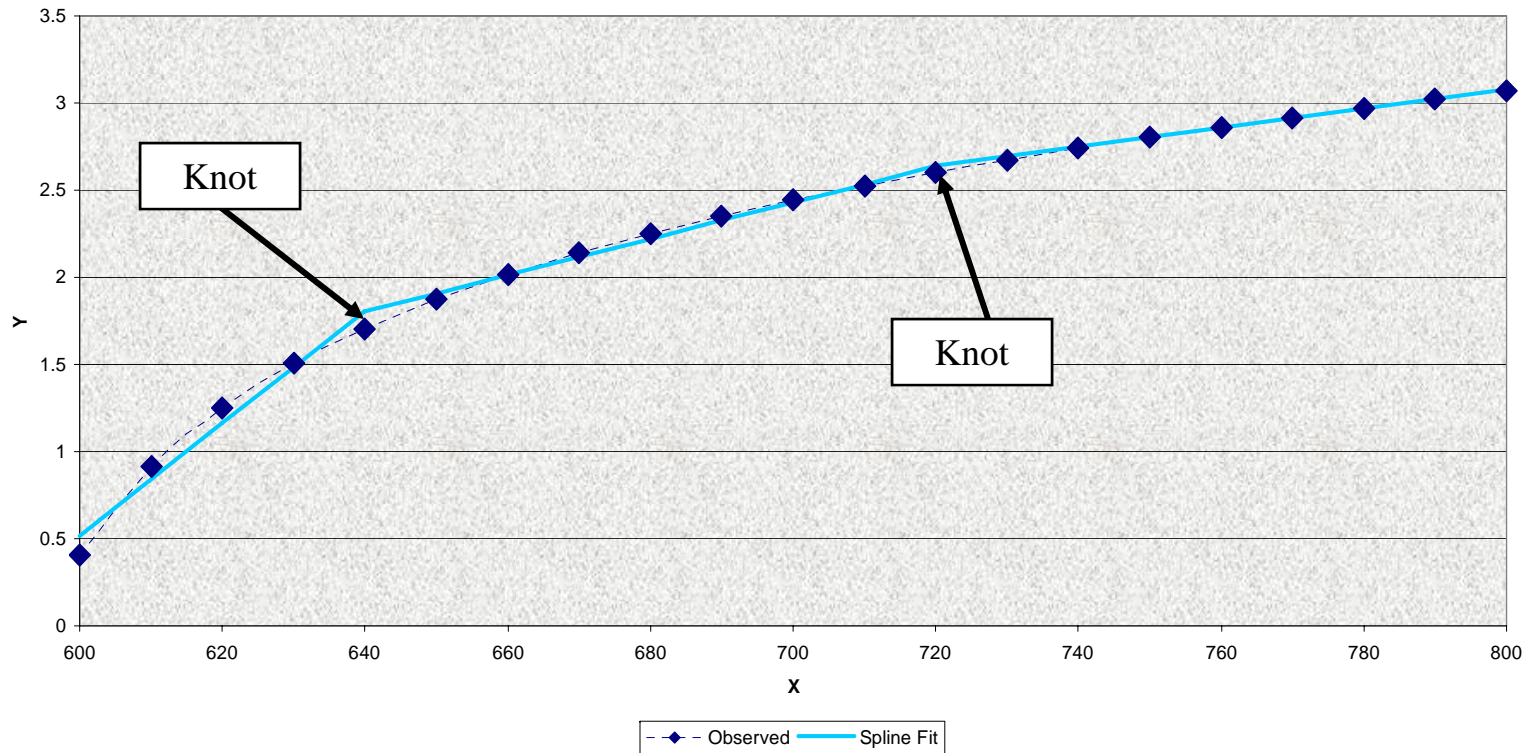


Spline remains continuous across all of X



Example 5 – Spline Data

Spline Example 5 - Three Piece Spline Data

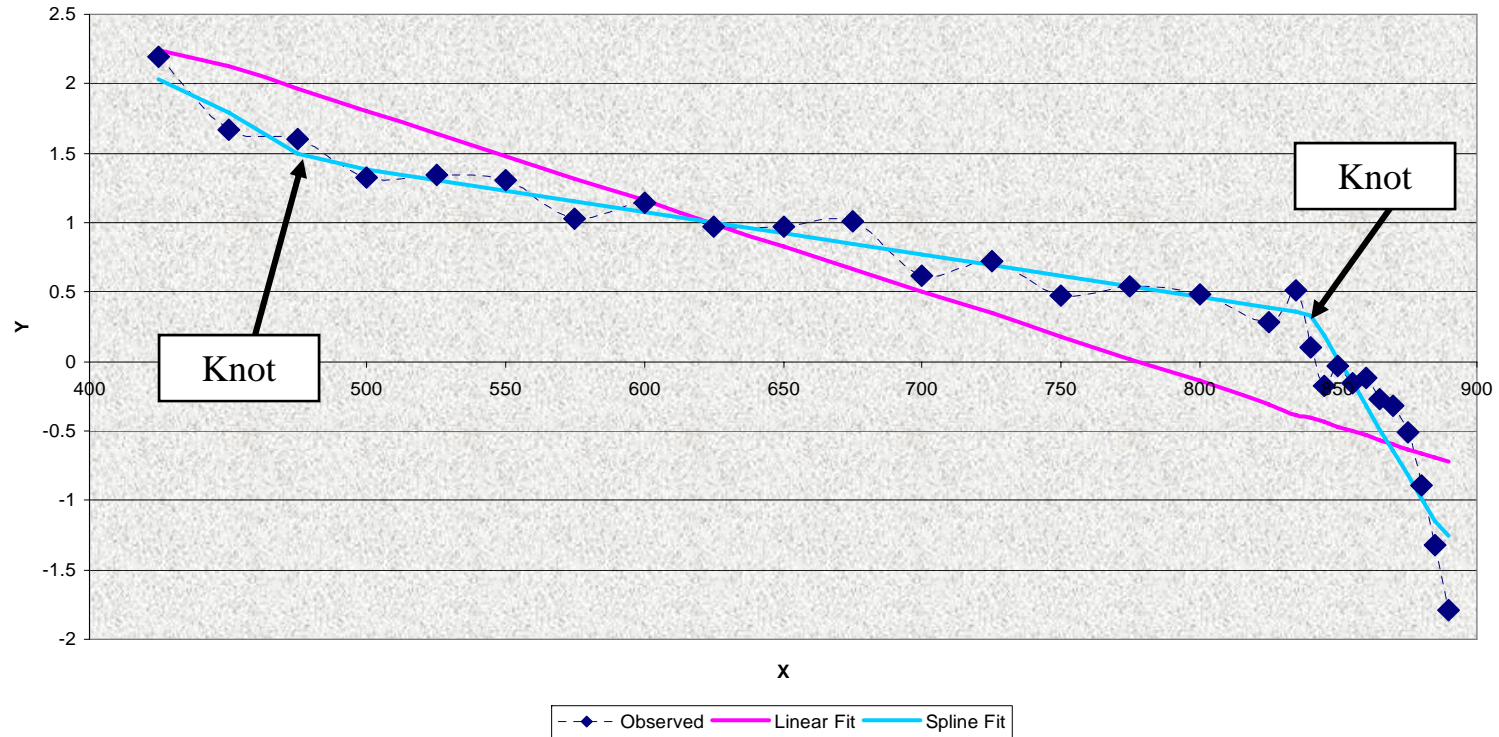


3 piece spline performs much better than 2 piece



Example 5 – Complicated Spline Data

Spline Example 5 - Spline Data



Model much better fit over X



How do we code in splines?

To create knots (e.g. 475), two methods can be used:

1. Relative Reference Coding

```
if x < 475 then x_spline_475 = 0;  
Else x_spline_475 = x - 475;
```

2. Absolute Reference Coding

```
if x < 475 then x_spline_475 = 475;  
Else x_spline_475 = x;
```

Methods differ only in interpretation for linear splines



Results From Logistic Regression

Relative Reference

Absolute Reference

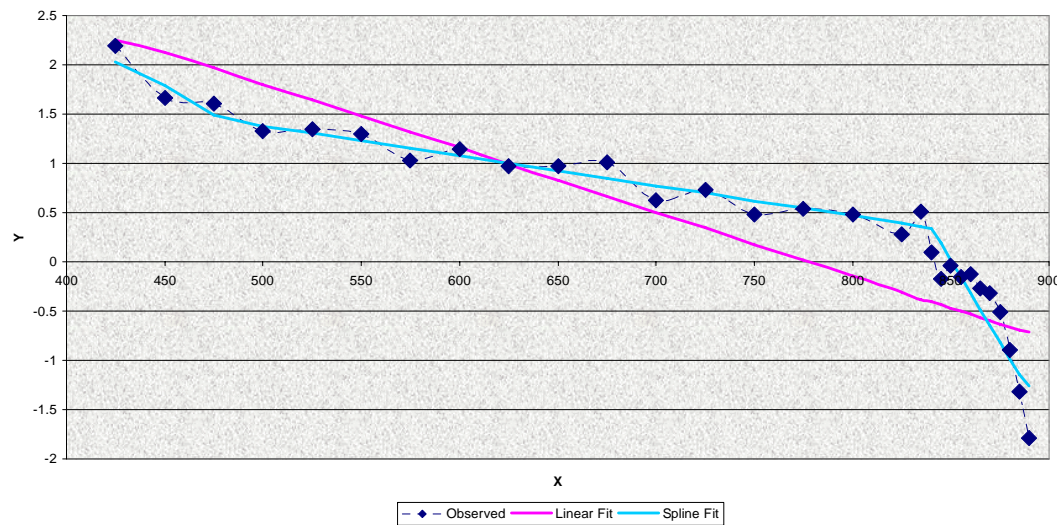
Analysis of Maximum Likelihood Estimates						Analysis of Maximum Likelihood Estimates					
Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq	Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	1	7.7601	1.4372	29.1531	<.0001	Intercept	1	28.3569	0.9112	968.5725	<.0001
x	1	-0.0133	0.00308	18.5567	<.0001	x	1	-0.0133	0.00308	18.5567	<.0001
spline_475	1	0.0102	0.00318	10.3743	0.0013	spline_475	1	0.0102	0.00318	10.3743	0.0013
spline_840	1	-0.0303	0.00118	660.2258	<.0001	spline_840	1	-0.0303	0.00118	660.2258	<.0001



Where do we place the knots?

Method 1 : Intuition

Spline Example 5 - Spline Data



	X alone	X + Splines
Model Fit Statistics	Intercept	Intercept
Criterion	Only	and
		Covariates
AIC	27,728	24,223
SC	27,736	24,238
-2 Log L	27,726	24,219



Method 2: Macro Approach

In pseudo-code:

- Loop through a range of points
- Set the spline values
- Run regression and save model fit
- Change spline points
- At the end, pick the “best fit” model



Version 1: Small Data – part 1

```
%macro recursive_spline_fit_short(datain, dataout,  
    x, y, spline_min, spline_max, iter);  
  
    data temp_xyzzy;  
        set &datain (keep=&x &y);  
        do s1_knot = &spline_min to (&spline_max - &iter) by &iter;  
            do s2_knot = s1_knot + &iter to &spline_max by &iter;  
                if &x < s1_knot then spline_1 = 0;  
                else spline_1 = &x - s1_knot;  
                if &x < s2_knot then spline_2 = 0;  
                else spline_2 = &x - s2_knot;  
                output;  
            end;  
        end;  
        keep &x &y spline_1 spline_2 s1_knot s2_knot;  
  
    proc sort data=temp_xyzzy;  
        by s1_knot s2_knot;  
  
    proc logistic data=temp_xyzzy outest=&dataout noprint;  
        by s1_knot s2_knot;  
        model &y = &x spline_1 spline_2;
```



Version 1: Small Data – part 2

...

```
proc sql;
select s1_knot, s2_knot
  into: fin_s1, :fin_s2
  from &dataout
  where _lnlike_ = (select max(_lnlike_) from &dataout);
quit;

proc logistic data=temp_xyzzy;
  model &y = &x spline_1 spline_2;
  where (s1_knot = &fin_s1) and (s2_knot = &fin_s2);
  title "s1_knot = &fin_s1 and s2_knot = &fin_s2";
run;
%mend;
```



Why Version 1 may cause you grief

The algorithm presented has a Big-O magnitude of $\text{Combin}(\text{points}, \text{knots})$

On a dataset with 20,000 records and 100 possible knot points, your temporary dataset will have almost 100,000,000 records!

Multiplication Factor	Knots						
	1	2	3	4	5	6	
P o i n t s	10	10	45	120	210	252	210
	20	20	190	1,140	4,845	15,504	38,760
	30	30	435	4,060	27,405	142,506	593,775
	40	40	780	9,880	91,390	658,008	3,838,380
	50	50	1,225	19,600	230,300	2,118,760	15,890,700
	60	60	1,770	34,220	487,635	5,461,512	50,063,860
	70	70	2,415	54,740	916,895	12,103,014	131,115,985
	80	80	3,160	82,160	1,581,580	24,040,016	300,500,200
	90	90	4,005	117,480	2,555,190	43,949,268	622,614,630
	100	100	4,950	161,700	3,921,225	75,287,520	1,192,052,400



Version 2: Large Data - 1st Stage

```
%macro spline_fit(datain, dataout, x, y, s1, s2, noprint=);
  %put &s1 &s2;
  data temp_xyzzy;
    set &datain (keep=&x &y);
    if &x < &s1 then spline_1 = 0;
    else spline_1 = &x - &s1;
    if &x < &s2 then spline_2 = 0;
    else spline_2 = &x - &s2;
    keep &x &y spline_1 spline_2;

  run;
  proc logistic data=temp_xyzzy outest=outest_xyzzy &noprint;
    model &y = &x spline_1 spline_2;

  run;
  data &dataout;
    s1_knot = &s1; s2_knot = &s2;
    set outest_xyzzy;

  run;

%mend;
```



Version 2: Large Data - 2nd Stage

```
%macro recursive_spline_fit(datain, dataout, x, y, spline_min, spline_max, iter);
  %IF %SYSFUNC (EXIST (work.semi_xyzzy)) = 1 %THEN %do;
    proc datasets lib = work nolist;
      delete semi_xyzzy;
    run;
  %end;
  %do s1 = &spline_min %to &spline_max - &iter %by &iter;
  %do s2 = &s1 + &iter %to &spline_max %by &iter;
    %spline_fit(&datain, hrm, &x, &y, &s1, &s2, noprint=noprint);
    proc append base = semi_xyzzy data=hrm;
  %end;
%end;
data &dataout; set semi_xyzzy; run;
proc logistic data=&datain;
  model &y = &x;
run;
proc sql;
select *
  from &dataout where _lnlike_ = (select max(_lnlike_) from &dataout);
select s1_knot, s2_knot
  into: fin_s1, :fin_s2
  from &dataout where _lnlike_ = (select max(_lnlike_) from &dataout);
quit;
%spline_fit(&datain, hrm, &x, &y, &fin_s1, &fin_s2);
%mend;
```



Results

```
%recursive_spline_fit(spline_test_5, spline_test_5_all, x, is_bad, 460, 875, 5);
```

Model Fit Statistics		
Criterion	Intercept Only	Intercept and Covariates
AIC	27727.887	24222.683
SC	27735.791	24238.490
-2 Log L	27725.887	24218.683

Analysis of Maximum Likelihood Estimates					
Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	1	-5.0453	0.1011	2490.5859	<.0001
x	1	0.00649	0.000126	2663.6698	<.0001

s1_def	s2_def	Link function	Convergence Status	Intercept: is_bad=0	x	spline_1	spline_2	Model Log Likelihood
830	875	LOGIT	0 Converged	-3.33081	0.003678	0.012131	0.078346	-11704.3



Results (cont.)

```
%recursive_spline_fit(spline_test_5, spline_test_5_all, x, is_bad, 460, 875, 5);
```

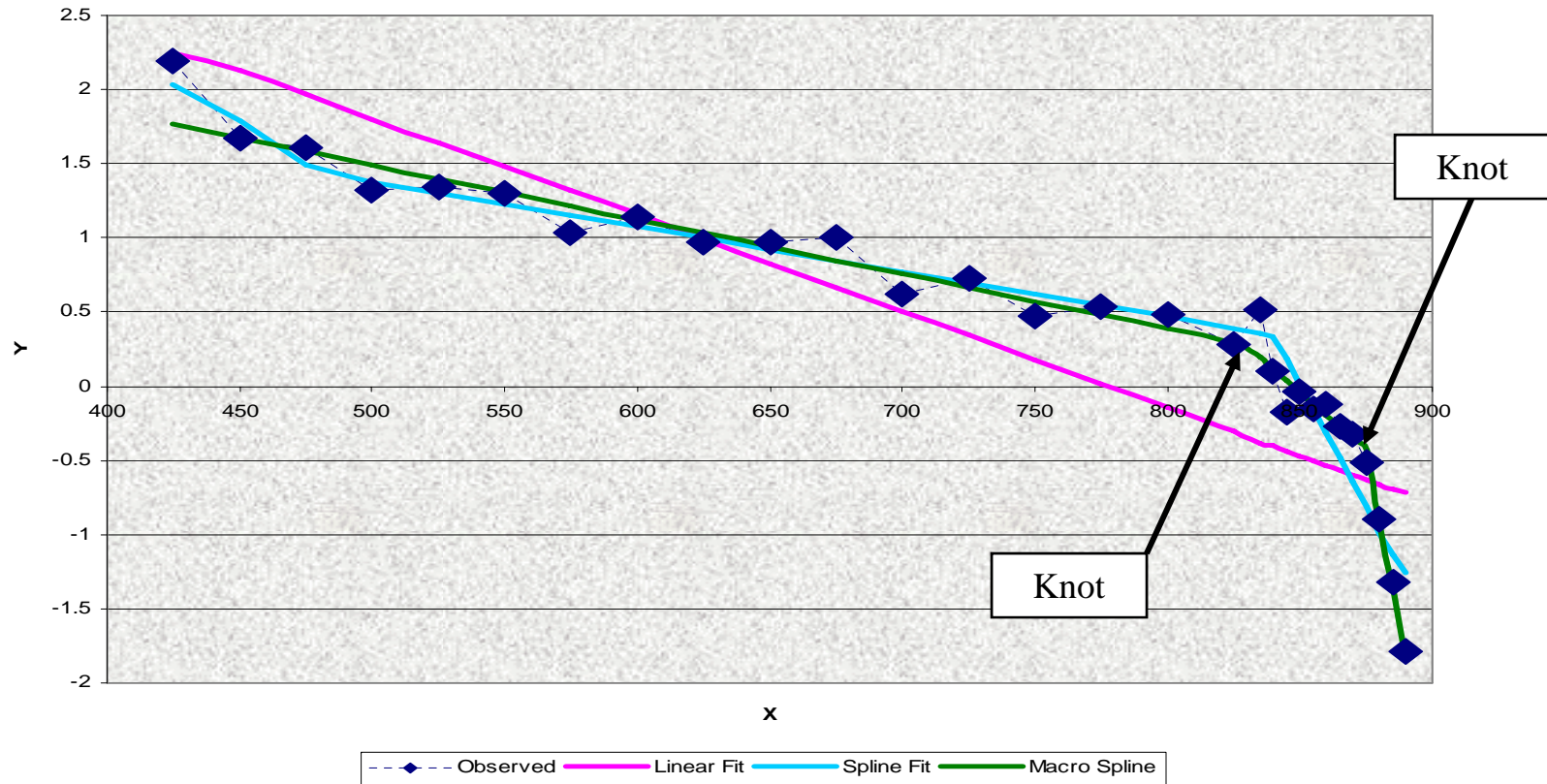
Model Fit Statistics		
Criterion	Intercept Only	Intercept and Covariates
AIC	27727.887	23416.602
SC	27735.791	23448.216
-2 Log L	27725.887	23408.602

Analysis of Maximum Likelihood Estimates					
Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	1	-3.3308	0.1222	743.4817	<.0001
x	1	0.00368	0.000177	432.8501	<.0001
spline_1	1	0.0121	0.00136	79.5160	<.0001
spline_2	1	0.0783	0.00623	158.3279	<.0001



Example 6 – Final Spline Data

Spline Example 6 - Final Data



Model sacrifices error on low values of X for better accuracy on high values of X



Conclusions

- Splines are an easy and convenient approach to modeling non-linear trends
- Macros can be used to loop through variables to determine optimal knots (and can be easily augmented for 3, 4, 5 knots!)
- Always pay attention to overfitting!





Questions?

Acknowledgements

Daymond Ling : spline tips

David Stone : debugging of SAS code

David L. Cassell : augmentation of macros

