Regressão linear e Ajuste de Curvas

Regressão Linear

Exemplo:

Dados o arquivo: *database/ExemploRegressaoLinear.txt*

y	x1	x2	x3
-22.26176	-6.884309	6.177759	-2.2524980
-17.90382	-2.612988	10.468689	-1.4931038
16.07747	8.919888	5.890522	-11.4111808
-18.33603	1.499722	14.512519	0.6867555
-25.66195	8.062538	27.865382	-7.8490362
-62.16295	5.034337	41.793461	-22.7948343
11.07620	8.699631	7.822655	25.1135465
-117.78906	5.757648	75.152487	22.2752464
-61.92541	1.473918	38.566852	-39.2139378
(….)			

Queremos ajustar um modelo linear para a variável dependente `y` a partir da observação das variáveis independentes: `x1`, `x2` e `x3`

Input e Output: Arquivo Texto

- Ler arquivos texto

Função genérica:

```
read.table(file, header = TRUE, sep = "\t", quote = "\"",
           dec = ".",fill = TRUE, comment.char = "",
           na.strings=c("#N/A N/A", ""),
           stringsAsFactors=FALSE...)
```

Função específica para ler arquivos com campos separados por 'Tab':

```
read.delim(file, dec=".",stringsAsFactors=FALSE,
           na.strings=c("#N/A N/A", ""));
```

- Escrever arquivos texto:

```
write.table(x, file = "", append = FALSE, quote = FALSE,
           sep = "\t", eol = "\n", na = "NA", dec = ".",
           row.names = FALSE, col.names = TRUE,
           qmethod = c("escape", "double"),
           fileEncoding = "")
```

Leitura de um Arquivo Texto

Análise do Arquivo:

`database/ExemploRegressaoLinear.txt` :

- O arquivo é separado por 'TAB': `sep = "\t"`
- O separador decimal é ",": `dec = ","`
- O arquivo tem *header*: `header = TRUE`
- Não vou usar strings com *factors*: `stringsAsFactors=FALSE`
- Qual a *string* utilizada para indicar dados Não Disponíveis?
- As strings estão *quoted*? R: Não.
- Há linhas comentadas? Qual o caractere? R: Não há comentários

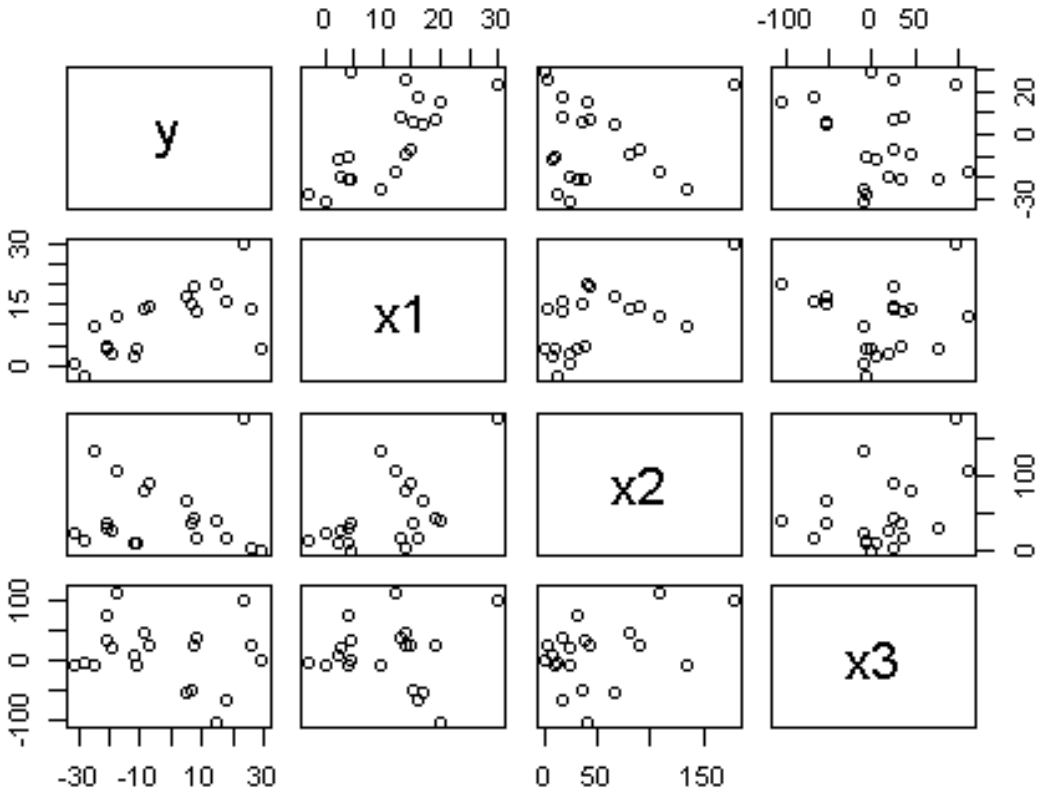
Regressão Linear

Ler os dados do arquivo:

```
> data=read.table("../database/ExemploRegressaoLinear.txt",
header=TRUE, sep="\t", dec=",");
```

Visualizar o comporta

```
> summary(data);
(...)
> pairs(data);
```



Regressão Linear

Formulação geral:

```
fitted.model <- lm(formula, data = data.frame)
```

Exemplo:

```
> fit = lm(y ~ x1 + x2 + x3, data=data);
> summary(fit)
```

```
Call:
lm(formula = y ~ x1 + x2 + x3, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-12.386  -5.000  -3.194   1.381  35.711

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -17.26214    4.24287  -4.069 0.000894 ***
x1            2.49935    0.40514   6.169 1.35e-05 ***
x2           -0.27347    0.07549  -3.623 0.002286 **
x3            0.02730    0.05484   0.498 0.625387
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 11.08 on 16 degrees of freedom
Multiple R-squared:  0.7165, Adjusted R-squared:  0.6633
F-statistic: 13.48 on 3 and 16 DF,  p-value: 0.0001206
```

Regressão Linear

Exemplo:

```
> fit = lm(y ~ x1 + log(x2), data=data);
> summary(fit)
```

```
Call:
lm(formula = y ~ x1 + log(x2), data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-4.7507 -0.9693  0.0732  1.4786  3.4303

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   5.34187    1.23066   4.341 0.000444 ***
x1            2.47301    0.06998  35.339  < 2e-16 ***
log(x2)     -10.84646    0.38813 -27.945 1.19e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.198 on 17 degrees of freedom
Multiple R-squared:  0.9881, Adjusted R-squared:  0.9868
F-statistic: 708.7 on 2 and 17 DF,  p-value: < 2.2e-16
```

Atributos de Objetos em R

A função `lm(.)` assim como outras funções em R, retorna um objeto que possui uma lista atributos.

- O atributos podem ser acessados diretamente ou através de funções específicas.
- Os atributos podem ser acessado como uma lista:

  `objeto$atributo`

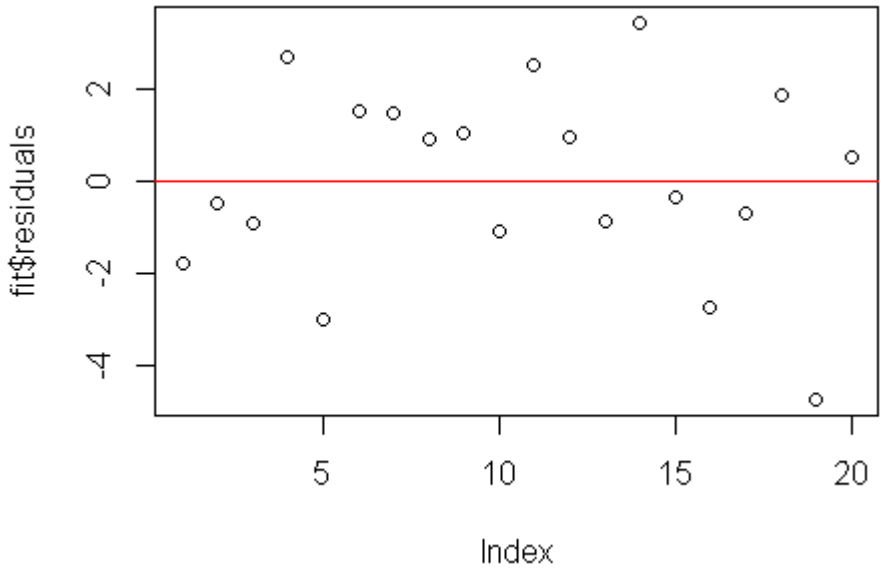
- Para saber quais são os atributos acessíveis do objeto possui utilizamos as funções: `names(objeto)` e `attributes(objeto)`

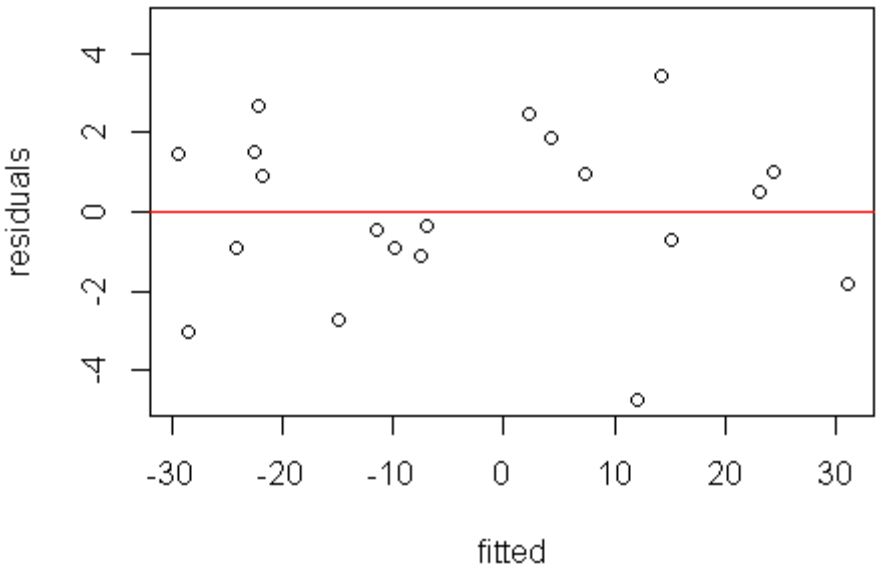
```
> names(fit)
[1] "coefficients" "residuals" "effects" "rank" "fitted.values" "assign"
[7] "qr" "df.residual" "xlevels" "call" "terms" "model"


> fit$coefficients
Intercept) x1 log(x2)
5.341865 2.473014 -10.846460
```

Atributos de Objetos em R

- Gráfico de Resíduo e Gráfico Valor Estimado

```
> plot(fit$residuals)
> abline(h=0,col="red")
> summary(fit$residuals)
      Min.  1st Qu.  Median  Mean   3rd  Qu.  Max.
-4.75100 -0.96930 0.07318 0.00000 1.47900 3.43000

> plot(fit$fitted.values, fit$residuals, xlab="fitted",
ylab="residuals",ylim=max(abs(fit$residuals))*c(-1,1))
> abline(h=0,col="red")
```

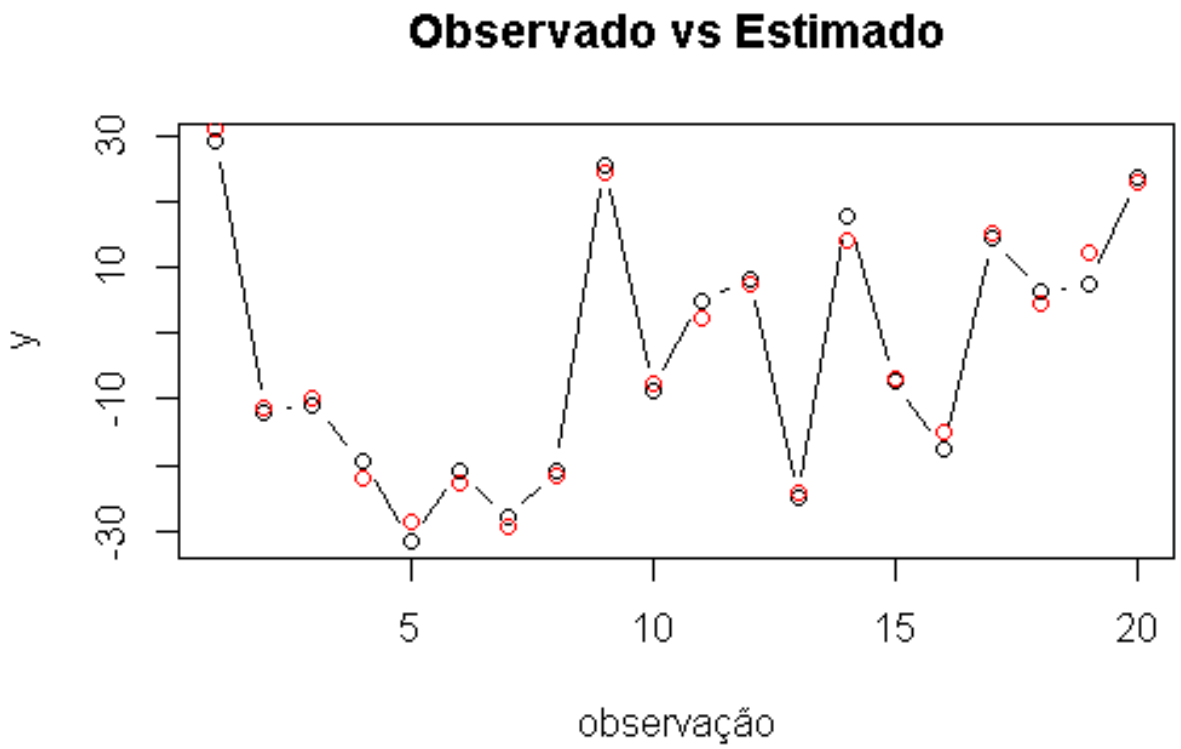




Atributos de Objetos em R

- Valor Observado vs Valor Estimado

```
> plot(data[['y']], type="b",
                    xlab="observação",
                    ylab="y",
                    main="Observado vs Estimado");
> points(fit$fitted.values, col="red");
```



Função `predict()` para modelos em R

Função `predict(...)` pode ser utilizado diversos modelos de previsão em R, como por exemplo, linear, polinomial e VAR.

- Sintaxe básica:

```
predict (object, ...)
```

Os demais argumentos, indicados por '....' varia de acordo com o modelo que o `object` representa.

No Exemplo:

- Carrega novos dados: 'out of sample'

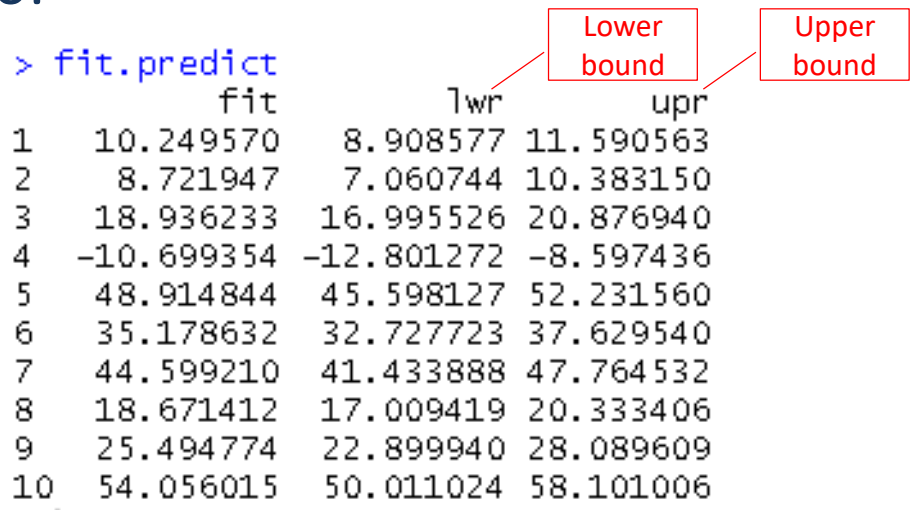
```
> newdata = read.table("../database/ExemploRegressaoLinear-
NewData.txt", header=TRUE, sep="\t", dec=",");
```

- Faz a previsão utilizando o modelo para os novos dados e retorna os valores para o intervalo de 95% de confiança

```
> fit.predict=predict(fit,newdata=newdata,interval="confidence",
level=.95);
```

Função `predict()` para modelos em R

Saída do resultado:




```
> fit.predict
          fit        lwr        upr
1   10.249570   8.908577 11.590563
2    8.721947   7.060744 10.383150
3   18.936233  16.995526 20.876940
4  -10.699354 -12.801272 -8.597436
5   48.914844  45.598127 52.231560
6   35.178632  32.727723 37.629540
7   44.599210  41.433888 47.764532
8   18.671412  17.009419 20.333406
9   25.494774  22.899940 28.089609
10  54.056015  50.011024 58.101006
```

Exemplo de utilização da função `matplot(x,y,...)` para vários gráficos:

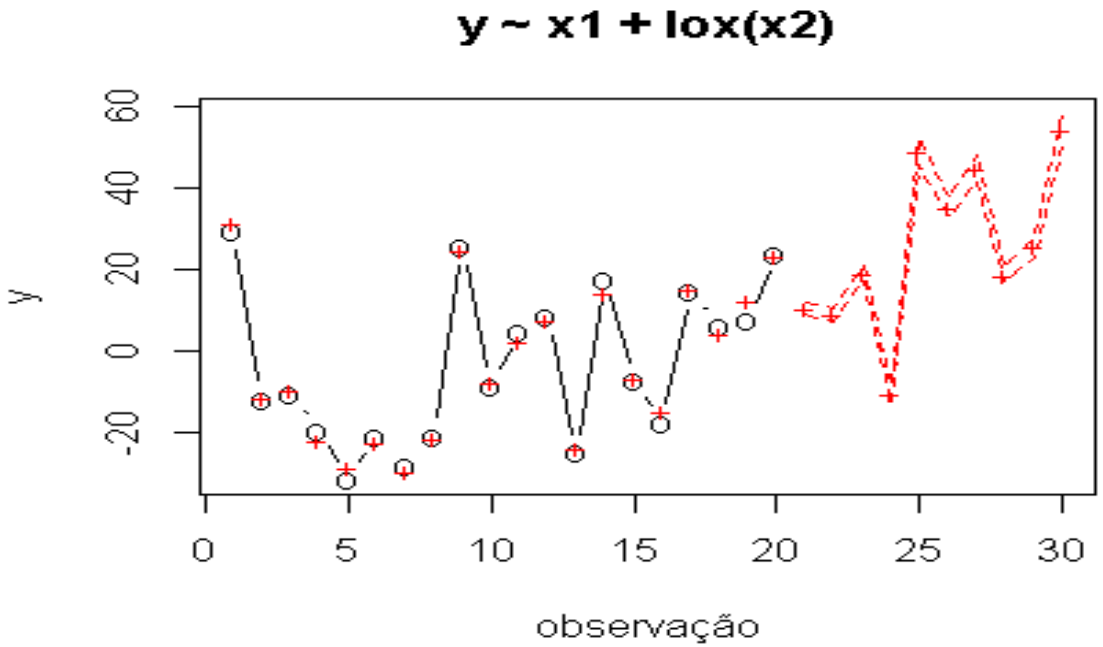
Prepara os dados a serem plotados:

```
alldata = data.frame( y = c(data[['y']], rep(NA,nrow(newdata))),
                      fitted = c(fit$fitted.values,fit.predict[,'fit']),
                      upper=c(rep(NA,nrow(data)),fit.predict[,'upr']),
                      lower=c(rep(NA,nrow(data)),fit.predict[,'lwr']));
```

Função `predict()` para modelos em R

Exemplo `matplot(x,y,...)`:

```
> matplot(1:nrow(alldata),as.matrix(alldata),
        type=c("b","p","l","l"),
        lty=c(1,1,2,2),
        pch=c("o","+"),
        col=c("black","red","red","red"),
        xlab="observação",
        ylab="y",
        main="y ~ x1 + lox(x2)");
```



Funções especiais para modelos `lm(...)`

`step(object)`

Testa vários modelo automaticamente e seleciona o melhor modelo a partir das variáveis informadas:

```
## Acrescenta a variável log_x2 aos dados
> data[['log_x2']] = log(data[['x2']]);
> fit.step = step(lm(y~.,data=data));
> summary(fit.step);
Call:
lm(formula = y ~ x1 + log_x2, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-4.7507 -0.9693  0.0732  1.4786  3.4303
(...)
```

A formula: '`y ~ .`' significa: "regredir `y` contra todas as demais variáveis"

Funções especiais para modelos `lm(...)`

`coef(fit)`

Extrai os coeficientes dos modelo. Equivalente a `fit$coefficients`

`residuals(fit)`

Extrai os coeficientes dos modelo. Equivalente a `fit$residuals`

`fitted(fit)`

Extrai os coeficientes dos modelo. Equivalente a `fit$fitted.values`

`formula(fit)`

Extrai os coeficientes dos modelo. Equivalente a `fit$call$formula`

`anova(fit_1, fit_2)`

Compara dois modelos e gera a tabela de análise de variância

Detalhes da Formula em R

Exemplos de fórmulas:

`y ~ x1` e `y ~ x1 + 1`

Regressão com intercepto

`y ~ -1 + x1`

Regressão SEM intercepto (passando pela origem)

`y ~ x1 + x2`

`log(y) ~ x1 + log(x2)`

Regressão com intercepto e duas variáveis e `log(.)`

`y ~ 1 + x + I(x^2) + I(x^3)`

Regressão com um polinômio de terceiro grau

`y ~ x1 * x2` ou `y ~ x1 + x2 + x1:x2`

As duas fórmulas indicam a mesma expressão. Efeito de cada variável isoladamente e o efeito cruzado: `x1, x2` e `x1*x2`

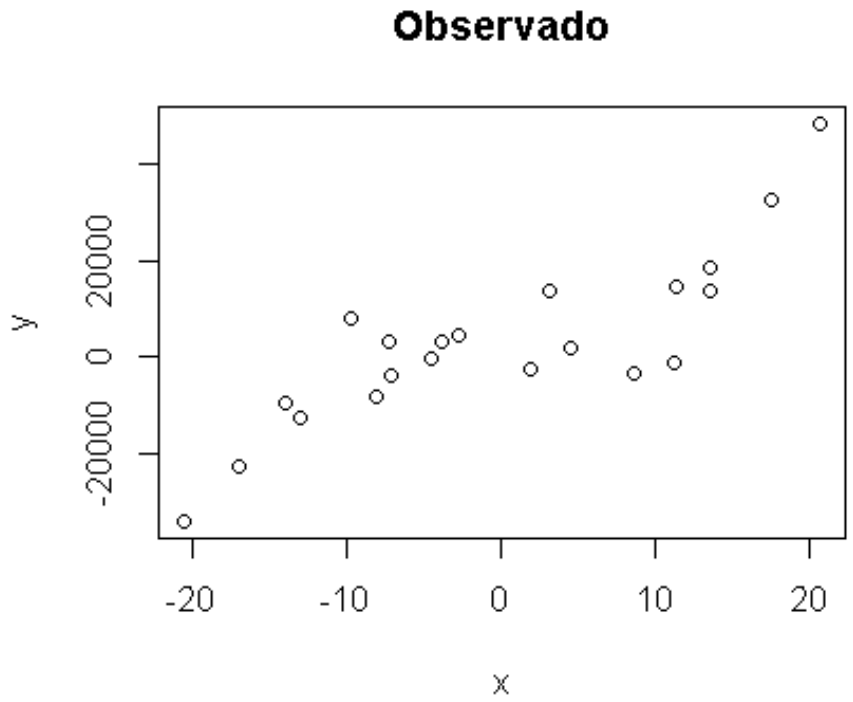
Ajuste Aproximado de Curvas - Polinômio

O ajuste aproximado de curvas pode ser feito com polinômios de grau utilizando a função `lm(.)` na forma:

```
y ~ 1 + x + I(x^2)+...+ I(x^n)
```

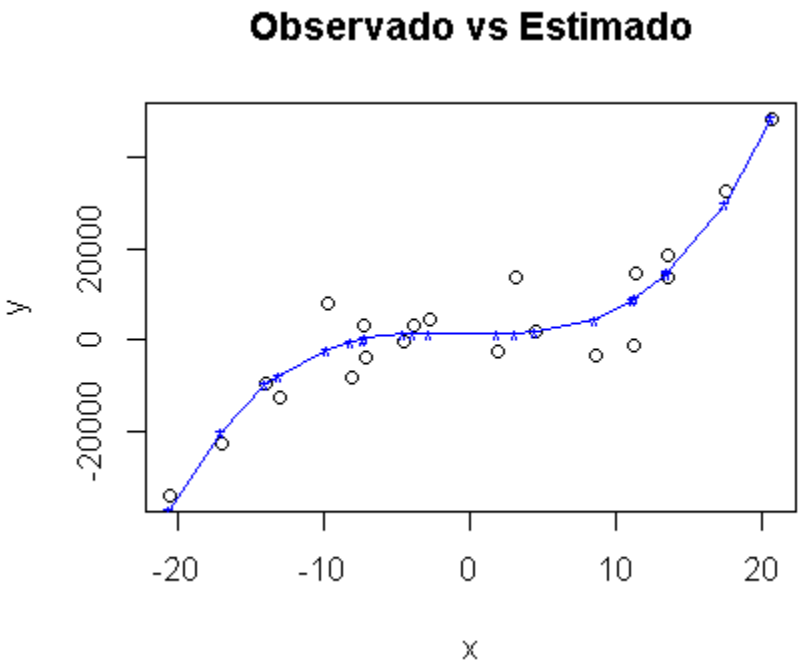
Exemplo:

Ajustar dos dados do arquivo: ExemploRegressaoPolinomio.txt utilizando um polinômio do terceiro grau



Ajuste Aproximado de Curvas - Polinômio

```
## Carrega a base de dados
data = read.table("../database/ExemploRegressaoPolinomio.txt", header=TRUE, sep="\t",
dec=",");

## Ajusta um polinômio de 3o grau
fit = lm(y~ x + I(x^2) + I(x^3),data=data);
summary(fit);

## Plota o valor Observado vs Estimado
plot(data[['x']], data[['y']], type="p",
    xlab="x",
    ylab="y",
    main="Observado vs Estimado");

lines(data[['x']], fit$fitted.values, col="blue");
points(data[['x']], fit$fitted.values, col="blue",pch="*");
```



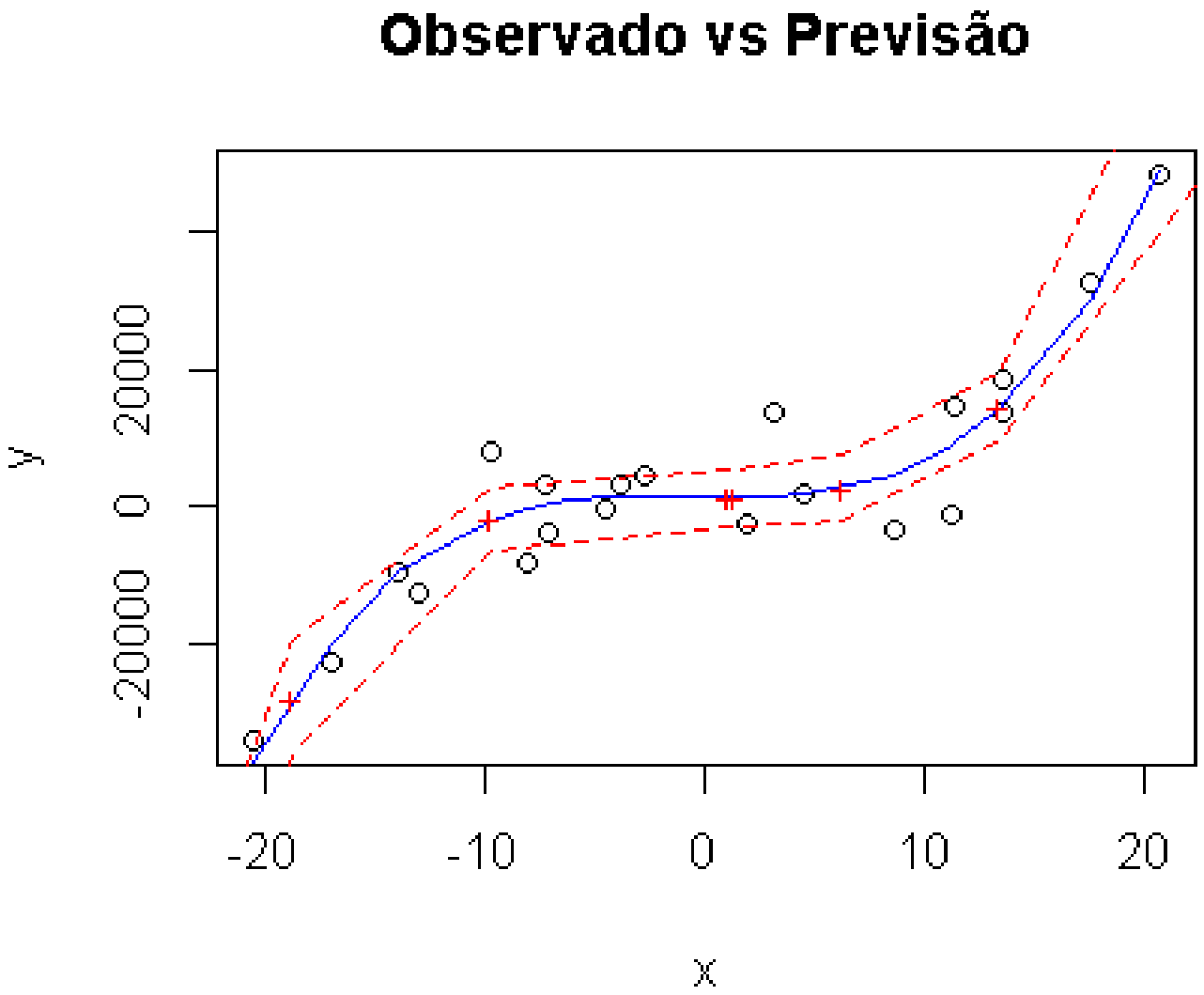
Ajuste Aproximado de Curvas - Polinômio

Utilizando a função `predict(.)` com o modelo ajustado:

```
## Carrega uma nova base de dados
>newdata = read.table("../database/ExemploRegressaoPolinomio-
NewData.txt.txt", header=TRUE, sep="\t", dec=",");

## Utiliza o modelo para previsão
>fit.predict=predict(fit,newdata=newdata,
                     interval="confidence",
                     level=.95;
## Plota os resultados
>plot( data[['x']],data[['y']],
       type="p",
       xlab="x", ylab="y",
       main="Observado vs Previsão");

>lines(data[['x']], fit$fitted.values, col="blue");
>points(newdata[['x']], fit.predict[,"fit"], col="red",pch="+");
>lines(newdata[['x']], fit.predict[,"upr"], col="red", lty=2);
>lines(newdata[['x']], fit.predict[,"lwr"], col="red", lty=2);
```

Ajuste Aproximado de Curvas - Polinômio

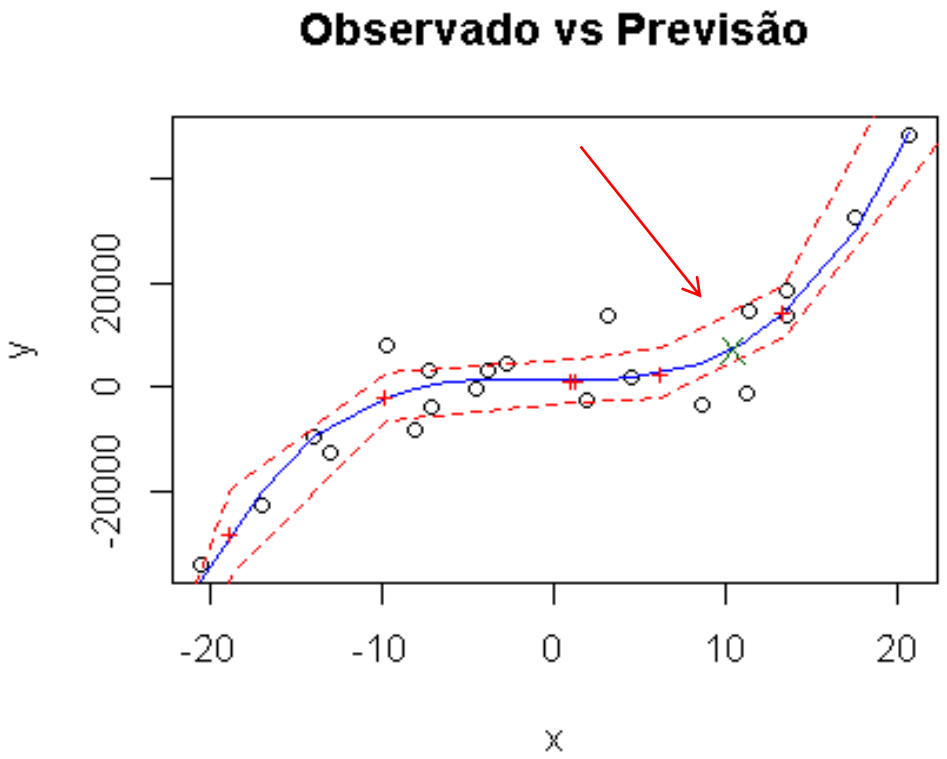
Utilizando a função `predict(.)` com o modelo ajustado:



Ajuste Aproximado de Curvas - Polinômio

Para interpolar um novo ponto qualquer basta utilizar os coeficientes e calcular o valor de y:

```
## Exemplo Interpolação
x_obs = 10.5;
y_obs = sum(coef(fit)*c(1,x_obs,x_obs^2,x_obs^3));
points(x_obs,y_obs, col="darkgreen", pch="X");
```



Ajuste Exato de Curvas - Splines

Para ajustar uma curva de forma exata - passando por todos os pontos da amostra - temos que utilizar uma curva parametrizável.

O curva mais comum é a Spline Cúbica:

```
splinefun(x, y = NULL, ...)
```

ou

```
spline(x, y = NULL, ...)
```

A funcão `splinefun(.)` retorna uma *`function`* que pode ser utilizada diretamente para ajustar novos pontos ou calcular interpolações.

Exemplo: Curva de Juros no Brasil

Vamos utilizar splines cúbicas para interpolar a estrutura a termo de juros no Brasil:

O dados estão na planilha: `BBG-CurvaDeJuros.xlsx [Juros]`

Exemplo de código utilizando RODBC:

```
library(RODBC);

## Connect to Excel
conn = odbcConnectExcel2007("../database/BBG-CurvaDeJuros.xlsx");
## Carrega os dados de uma planilha
data = sqlFetch(conn, "Juros", na.strings=c(""));
## Close connection
odbcClose(conn);

## Taxa de juros
yields = data[['Taxa']];

## Dias úteis
wdays= data[['DU']];
```

Exemplo: Curva de Juros no Brasil

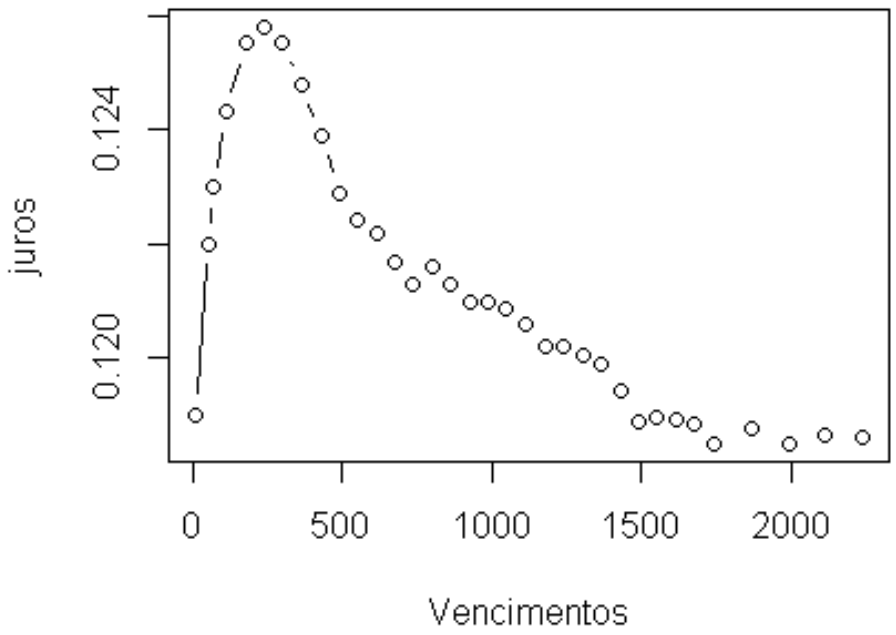
Exemplo de código utilizando XLConnect:

```
library(XLConnect);

## Carrega o workbook
wb = loadWorkbook("../database/BBG-CurvaDeJurios-2.xlsx");

## Atenção - para ler planilhas com fórmulas utilizar a opção
## "useCachedValues=TRUE" caso contrário XLConnect tenta fazer
## eval da fórmula e como está conectado na BBG dá erro!!!
data = readWorksheet(wb,
                     "juros",
                     startRow=1,startCol=1,
                     header=TRUE,
                     useCachedValues=TRUE);

## Taxa de juros
yields = data[['Taxa']];

## Dias úteis
wdays= data[['DU']];
```

Exemplo: Curva de Juros no Brasil

Ajuste da Spline

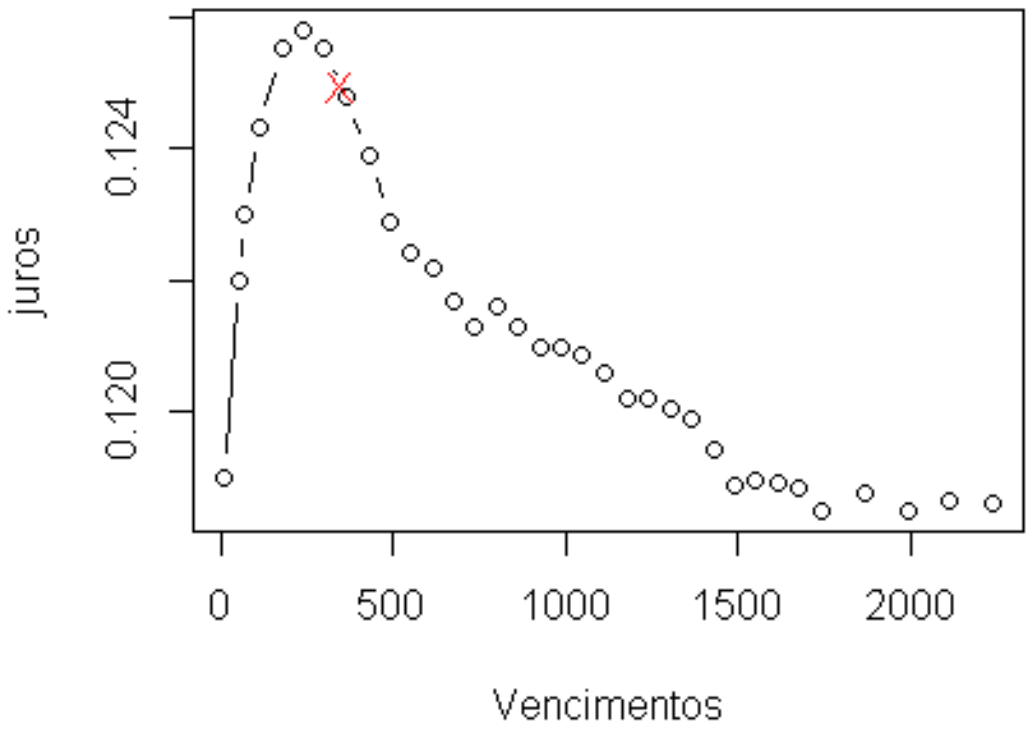
```
## Ajusta a função spline
yieldCurve = splinefun(wdays,yields);

## Plota o gráfico
plot(wdays,yieldCurve(wdays),
type="b",ylab="juros",xlab="Vencimentos")
```



Exemplo: Curva de Juros no Brasil

Podem os utilizar a função para interpolar pontos na Spline:

```
## Exemplo de interpolação
wday_obs = 350;
yield_obs = yieldCurve(wday_obs)
print(yield_obs)
points(wday_obs,yield_obs,pch="X",col="red");
```



Exercício 01

Carregar a base de dados : BBG-FX-Daily.txt

- Tratamento dos dados:
  - Selecionar os dados posteriores a 2008-01-01
  - Dividir os dados percentuais por 100:

`USDBRL25R1M, USDBRLV1M, BCSWFPD, BCSWCPD, GT10, USSA1`

  - Criar a variável: `DSWAP1YBR_US = BCSWCPD – USSA1`
  - Calcular a média mensal para cada variável
  - Eliminar as colunas nas quais ao menos um valor seja '*NaN*'
  - Separar a amostra em duas
    - Training: anterior a 2011-01-01
    - Predict: posterior a 2011-01-01
  - Fazer a regressão:

`BRL~USDBRL25R1M+USDBRLV1M+DXY+CRY+USSA1+DSWAP1YBR_US`

  - Fazer a regressão: selecionando o melhor modelo com `step(.)`

Exercício 02: Janela Móvel

Carregar a base de dados : BBG-FX-Daily.txt

- Tratamento dos dados:
  - Selecionar os dados posteriores a 2006-01-01
  - Dividir os dados percentuais por 100:

    `USDBRL25R1M, USDBRLV1M, BCSWFPD, BCSWCPD, GT10, USSA1`
  - Criar a variável: `DSWAP1YBR_US = BCSWCPD – USSA1`
  - Calcular a média SEMANAL para cada variável
  - Eliminar as colunas nas quais ao menos um valor seja '*NaN*'
  - Utilizar uma janela móvel de 78 semanas
    - Training: semana 1 a 52
    - Predict: 53 a 78
  - Fazer a regressão: selecionando o melhor modelo com `step(.)`

Exercício: Comparação do Poder de Previsão (cont...)

Exercício 02 (continuação)

Erxercicio com previsão usando janela móvel + trabalhar com primeiras diferenças e retornos