Regression

Anja Bråthen Kristoffersen

Biomedical Research Group

UNIVERSITY OF OSLO

http://cran.r-project.org/doc/contrib/Short-refcard.pdf

R Reference Card

R Reference Card

by Tom Short, EPRI PEAC, tshort@epri-peac.com 2004-11-07

Getting help

Most R functions have online documentation.

`help(topic)` documentation on `topic`
`?topic` id.
`help.search("topic")` search the help system
`apropos("topic")` the names of all objects in the search list matching the regular expression "topic"
`help.start()` start the HTML version of help
`str(a)` display the internal *str*ucture of an R object
`summary(a)` gives a "summary" of `a`, usually a statistical summary but it is *generic* meaning it has different operations for different classes of `a`
`ls()` show objects in the search path; specify `pat="pat"` to search on a pattern
`ls.str()` str() for each variable in the search path
`dir()` show files in the current directory
`methods(a)` shows S3 methods of `a`
`methods(class=class(a))` lists all the methods to handle objects of class `a`

Input and output

`load()` load the datasets written with `save`
`data(x)` loads specified data sets
`library(x)` load add-on packages
`read.table(file)` reads a file in table format and creates a data frame from it; the default separator `sep=""` is any whitespace; use `header=TRUE` to read the first line as a header of column names; use `as.is=TRUE` to prevent character vectors from being converted to factors; use `comment.char=""` to prevent `"#"` from being interpreted as a comment; use `skip=n` to skip `n` lines before reading data; see the help for options on row naming, NA treatment, and others
`read.csv("filename",header=TRUE)` id. but with defaults set for reading comma-delimited files
`read.delim("filename",header=TRUE)` id. but with defaults set for reading tab-delimited files
`read.fwf(file,widths,header=FALSE,sep="",as.is=FALSE)` read a table of *f*ixed *w*idth *f*ormatted data into a 'data.frame'; `widths` is an integer vector, giving the widths of the fixed-width fields
`save(file,...)` saves the specified objects (...) in the XDR platform-independent binary format
`save.image(file)` saves all objects
`cat(..., file="", sep=" ")` prints the arguments after coercing to character; `sep` is the character separator between arguments
`print(a, ...)` prints its arguments; generic, meaning it can have different methods for different objects
`format(x,...)` format an R object for pretty printing
`write.table(x,file="",row.names=TRUE,col.names=TRUE, sep=" ")` prints `x` after converting to a data frame; if `quote` is `TRUE`, character or factor columns are surrounded by quotes ("); `sep` is the field separator; `eol` is the end-of-line separator; `na` is the string for missing values; use `col.names=NA` to add a blank column header to get the column headers aligned correctly for spreadsheet input
`sink(file)` output to `file`, until `sink()`

Most of the I/O functions have a `file` argument. This can often be a character string naming a file or a connection. `file=""` means the standard input or output. Connections can include files, pipes, zipped files, and R variables.

On windows, the file connection can also be used with `description = "clipboard"`. To read a table copied from Excel, use
`x <- read.delim("clipboard")`
To write a table to the clipboard for Excel, use
`write.table(x,"clipboard",sep="\t",col.names=NA)`
For database interaction, see packages `RODBC`, `DBI`, `RMySQL`, `RPgSQL`, and `ROracle`. See packages `XML`, `hdf5`, `netCDF` for reading other file formats.

Data creation

`c(...)` generic function to combine arguments with the default forming a vector; with `recursive=TRUE` descends through lists combining all elements into one vector
`from:to` generates a sequence; ":" has operator priority; 1:4 + 1 is "2,3,4,5"
`seq(from,to)` generates a sequence `by=` specifies increment; `length=` specifies desired length
`seq(along=x)` generates `1, 2, ..., length(along)`; useful for `for` loops
`rep(x,times)` replicate `x` `times`; use `each=` to repeat "each" element of `x` `each` times; `rep(c(1,2,3),2)` is `1 2 3 1 2 3`; `rep(c(1,2,3),each=2)` is `1 1 2 2 3 3`
`data.frame(...)` create a data frame of the named or unnamed arguments; `data.frame(v=1:4,ch=c("a","B","c","d"),n=10);` shorter vectors are recycled to the length of the longest
`list(...)` create a list of the named or unnamed arguments; `list(a=c(1,2),b="hi",c=3i);`
`array(x,dim=)` array with data `x`; specify dimensions like `dim=c(3,4,2)`; elements of `x` recycle if `x` is not long enough
`matrix(x,nrow=,ncol=)` matrix; elements of `x` recycle
`factor(x,levels=)` encodes a vector `x` as a factor
`gl(n,k,length=n*k,labels=1:n)` generate levels (factors) by specifying the pattern of their levels; `k` is the number of levels, and `n` is the number of replications
`expand.grid()` a data frame from all combinations of the supplied vectors or factors
`rbind(...)` combine arguments by rows for matrices, data frames, and others
`cbind(...)` id. by columns

Slicing and extracting data

Indexing vectors

`x[n]`	n^{th} element
`x[-n]`	all *but* the n^{th} element
`x[1:n]`	first `n` elements
`x[-(1:n)]`	elements from `n+1` to the end
`x[c(1,4,2)]`	specific elements
`x["name"]`	element named `"name"`
`x[x > 3]`	all elements greater than 3
`x[x > 3 & x < 5]`	all elements between 3 and 5
`x[x %in% c("a","and","the")]`	elements in the given set

Indexing lists

`x[n]`	list with elements `n`
`x[[n]]`	n^{th} element of the list
`x[["name"]]`	element of the list named `"name"`
`x$name`	id.

Indexing matrices

`x[i,j]`	element at row `i`, column `j`
`x[i,]`	row `i`
`x[,j]`	column `j`
`x[,c(1,3)]`	columns 1 and 3
`x["name",]`	row named `"name"`

Indexing data frames (matrix indexing plus the following)

`x[["name"]]`	column named `"name"`
`x$name`	id.

Variable conversion

`as.array(x)`, `as.data.frame(x)`, `as.numeric(x)`, `as.logical(x)`, `as.complex(x)`, `as.character(x)`, ... convert type; for a complete list, use `methods(as)`

Variable information

`is.na(x)`, `is.null(x)`, `is.array(x)`, `is.data.frame(x)`, `is.numeric(x)`, `is.complex(x)`, `is.character(x)`, ... test for type; for a complete list, use `methods(is)`
`length(x)` number of elements in `x`
`dim(x)` Retrieve or set the dimension of an object; `dim(x) <- c(3,2)`
`dimnames(x)` Retrieve or set the dimension names of an object
`nrow(x)` number of rows; `NROW(x)` is the same but treats a vector as a one-row matrix
`ncol(x)` and `NCOL(x)` id. for columns
`class(x)` get or set the class of `x`; `class(x) <- "myclass"`
`unclass(x)` remove the class attribute of `x`
`attr(x,which)` get or set the attribute `which` of `x`
`attributes(obj)` get or set the list of attributes of `obj`

Data selection and manipulation

`which.max(x)` returns the index of the greatest element of `x`
`which.min(x)` returns the index of the smallest element of `x`
`rev(x)` reverses the elements of `x`
`sort(x)` sorts the elements of `x` in increasing order; to sort in decreasing order: `rev(sort(x))`
`cut(x,breaks)` divides `x` into intervals (factors); `breaks` is the number of cut intervals or a vector of cut points
`match(x, y)` returns a vector of the same length than `x` with the elements of `x` which are in `y` (`NA` otherwise)
`which(x == a)` returns a vector of the indices of `x` if the comparison operation is true (`TRUE`), in this example the values of `i` for which `x[i] == a` (the argument of this function must be a variable of mode logical)
`choose(n, k)` computes the combinations of k events among n repetitions $= n!/[(n-k)!k!]$
`na.omit(x)` suppresses the observations with missing data (`NA`) (suppresses the corresponding line if `x` is a matrix or a data frame)
`na.fail(x)` returns an error message if `x` contains at least one `NA`

Simple linear regression

- Describes the relationship between two variables x and y

$$y = \alpha + \beta x + \varepsilon$$

- The numbers α and β are called parameters, and ϵ is the error term.

Example data

- Waiting **time between eruptions** and the **duration** of the eruption for the Old Faithful geyser in Yellowstone National Park, Wyoming, USA.

head(faithful)

	eruptions	waiting
1	3.600	79
2	1.800	54
3	3.333	74
4	2.283	62
5	4.533	85
6	2.883	55

```
> eruption.lm = lm(eruptions ~ waiting, data=faithful)
> summary(eruption.lm)

Call:
lm(formula = eruptions ~ waiting, data = faithful)

Residuals:
     Min       1Q   Median       3Q      Max
-1.29917 -0.37689  0.03508  0.34909  1.19329

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.874016   0.160143  -11.70   <2e-16 ***
waiting      0.075628   0.002219   34.09   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4965 on 270 degrees of freedom
Multiple R-squared: 0.8115,     Adjusted R-squared: 0.8108
F-statistic:  1162 on 1 and 270 DF,  p-value: < 2.2e-16
```

Estimated parameters

P-value

Coefficient of Determination

Coefficient of determination

- the quotient of the variances of the **fitted values** and **observed values** of the dependent variable.

$$r^2 = \frac{\sum (\hat{y}_i - \bar{y})^2}{\sum (y_i - \bar{y})^2}$$

Prediction

- develop a 95% confidence interval of the mean eruption duration for the waiting time of 80 minutes

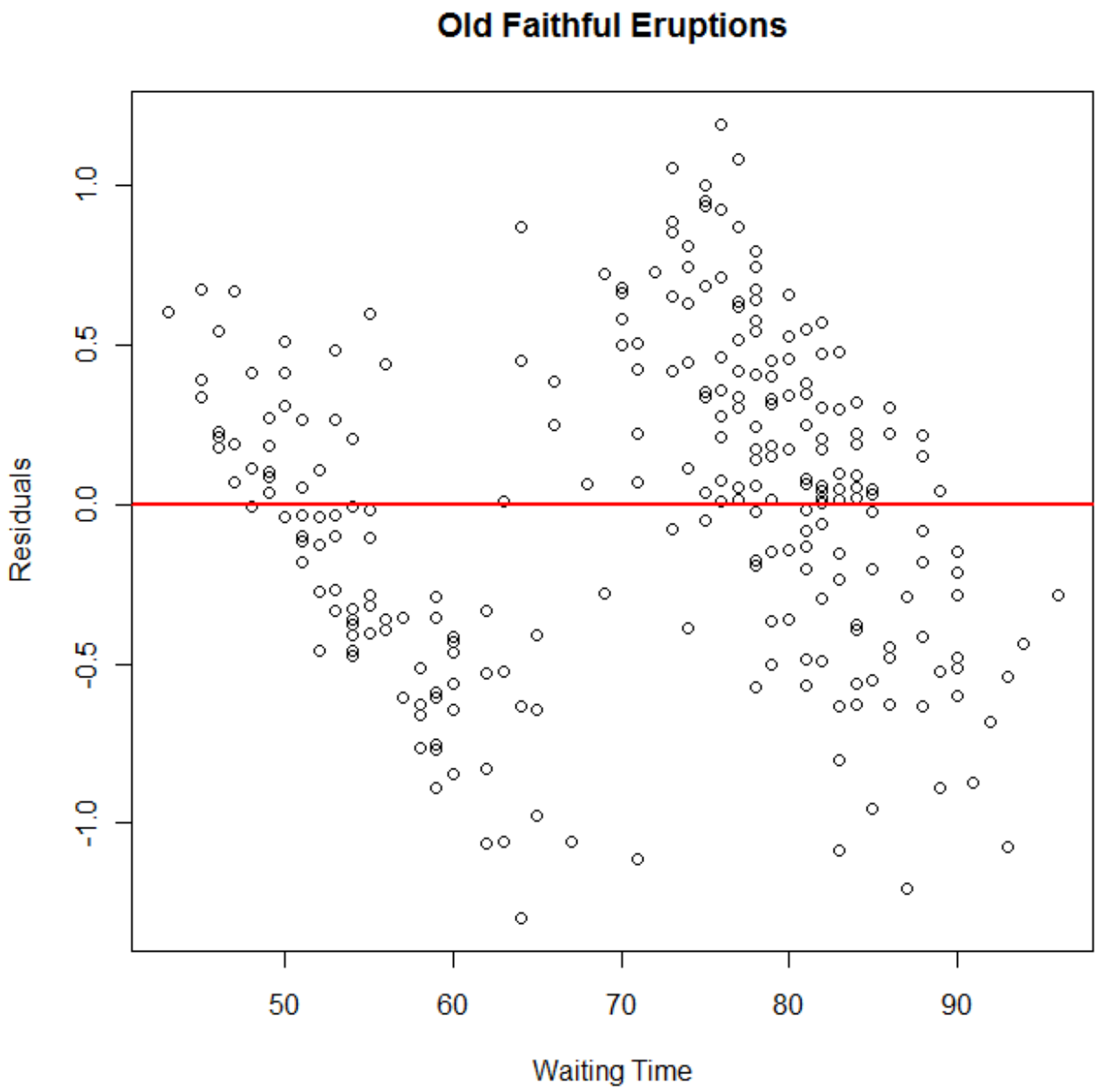
```
newdata <- data.frame(waiting=80)
predict(eruption.lm, newdata, interval="confidence")
       fit      lwr      upr
1 4.1762 4.1048 4.2476
```

Residuals

- The difference between the observed data of the dependent variable y and the fitted values ŷ.

$$Residual = y - \hat{y}$$

```
eruption.res <- resid(eruption.lm)
plot(faithful$waiting, eruption.res, ylab="Residuals",
     xlab="Waiting Time", main="Old Faithful Eruptions")
abline(0,0, col = "red", lwd = 2)
```

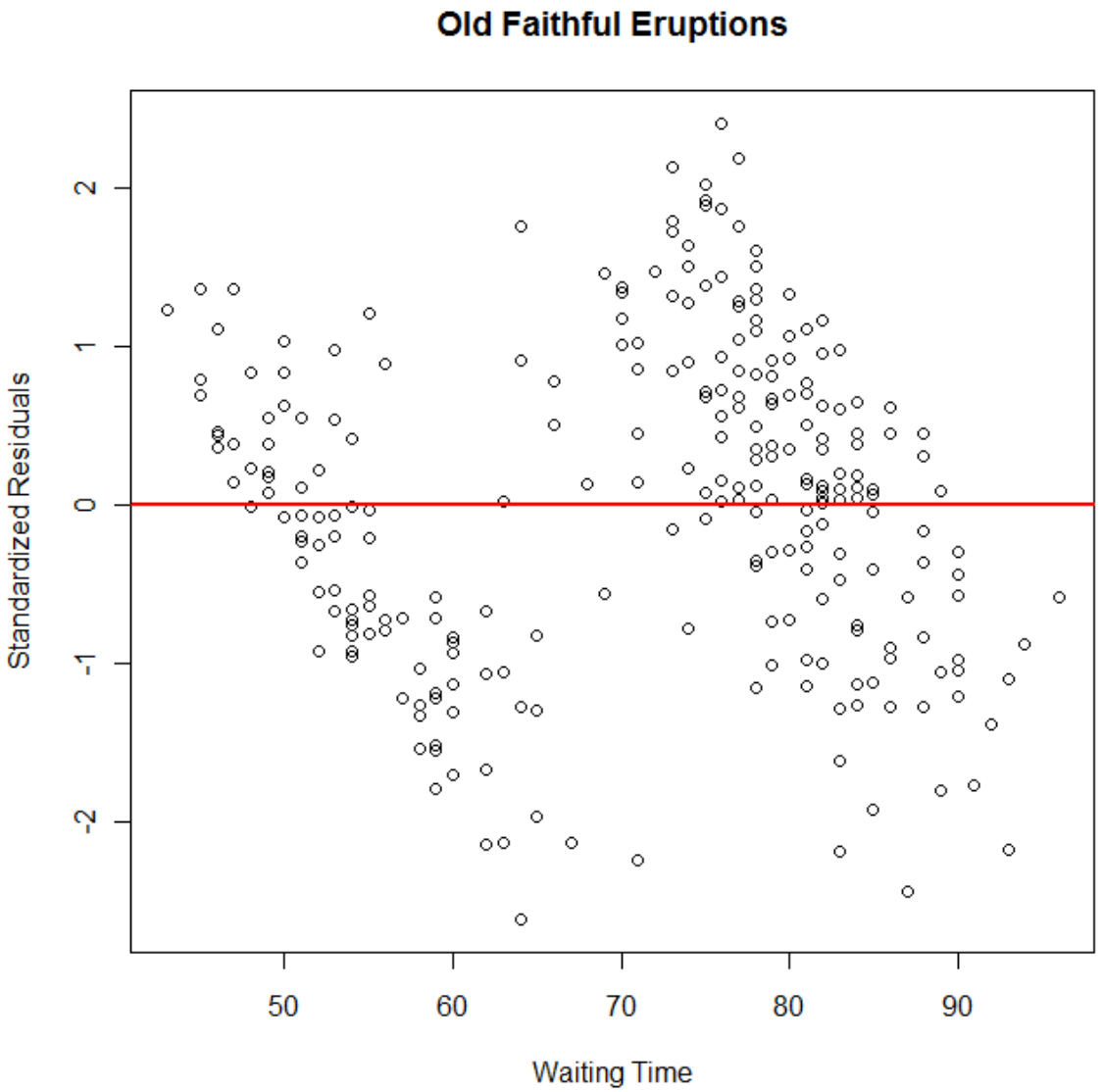


Standardized residual

- the residual divided by its standard deviation

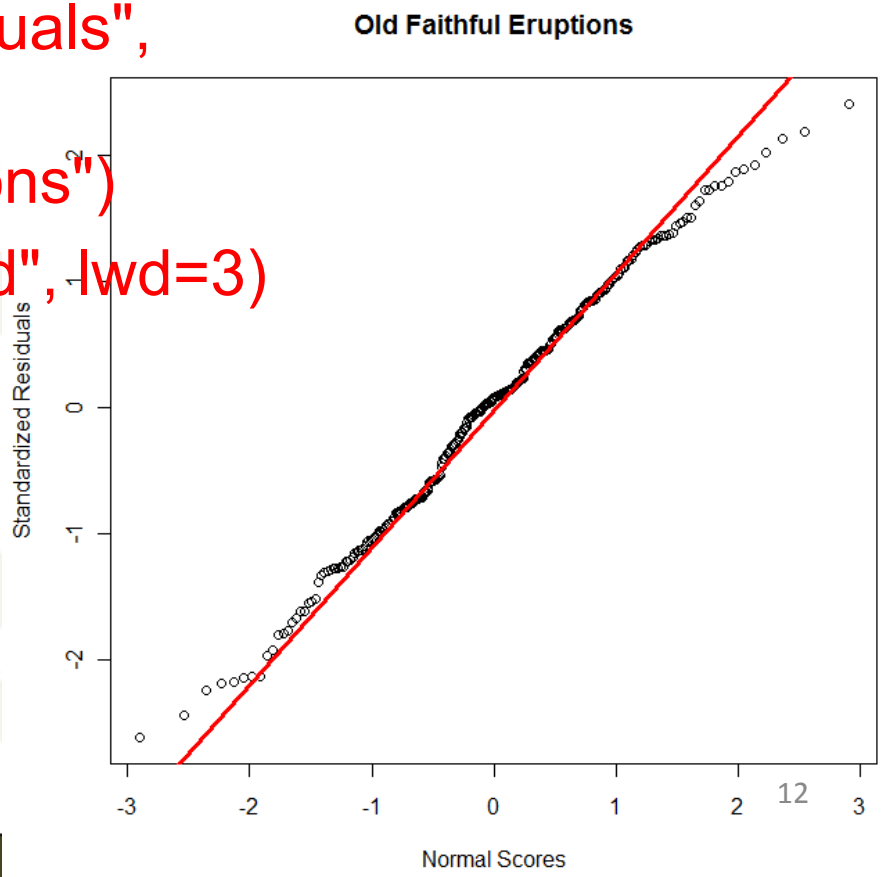
$$Standardized\ residual_i = \frac{residual_i}{standard\ deviation\ of\ residual_i}$$

```
eruption.stdres = rstandard(eruption.lm)
plot(faithful$waiting, eruption.stdres,
    ylab="Standardized Residuals", xlab="Waiting Time",
    main="Old Faithful Eruptions")
abline(0, 0, col ="red", lwd = 2)
```



Normal Probability Plot of Residuals

```
qqnorm(eruption.stdres,
       ylab="Standardized Residuals",
       xlab="Normal Scores",
       main="Old Faithful Eruptions")
qqline(eruption.stdres, col = "red", lwd=3)
```



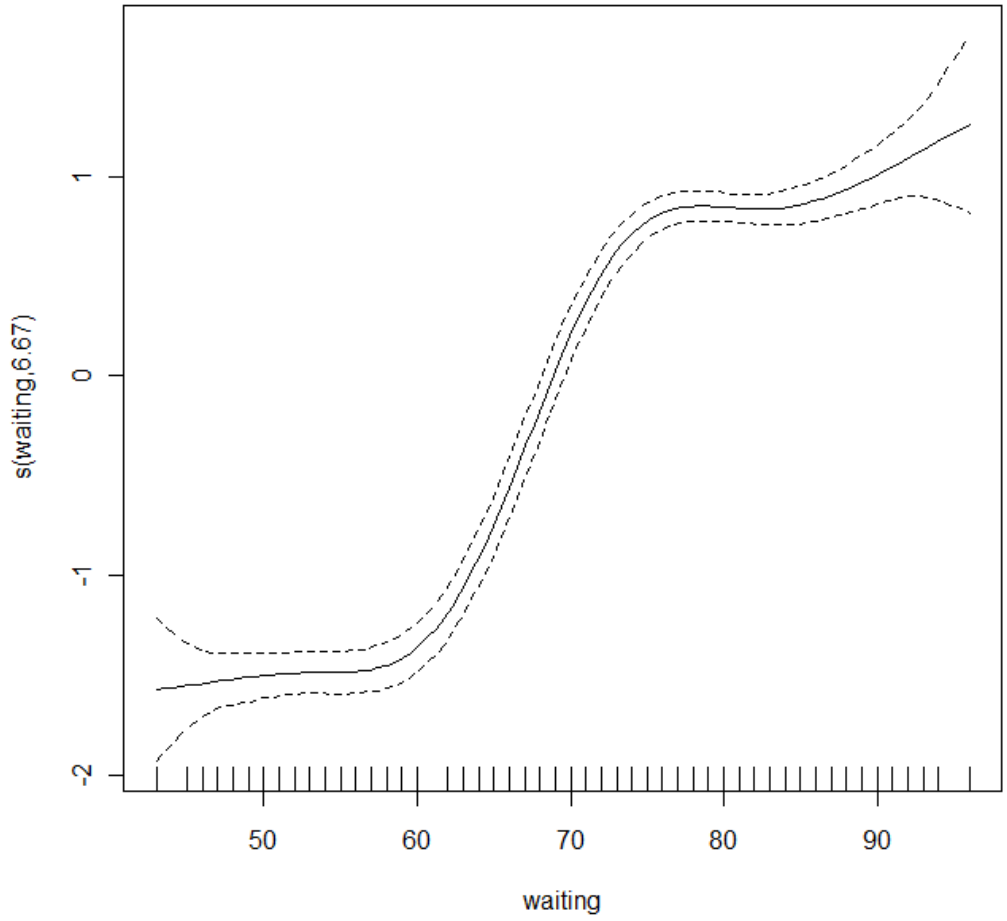
Generalized additive model

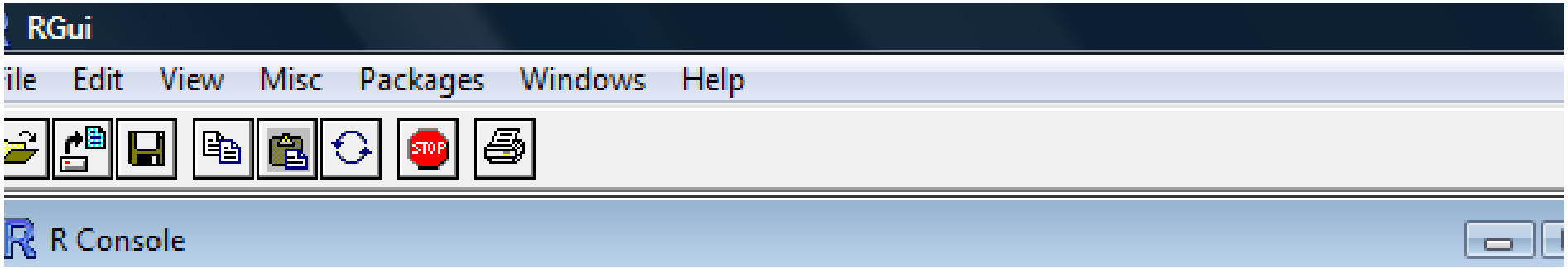
- Can replace the linear relationship between response and variable, as in linear regression, with a non-linear relationship. A spline.

GAM

```
install.packages("mgcv")
library(mgcv)
This is mgcv  1.5-5 . For overview type
`help("mgcv-package")'.
eruption.gam <- gam(eruptions ~ 1+s(waiting),
   data=faithful)
plot(eruption.gam)
```

plot(eruption.gam)



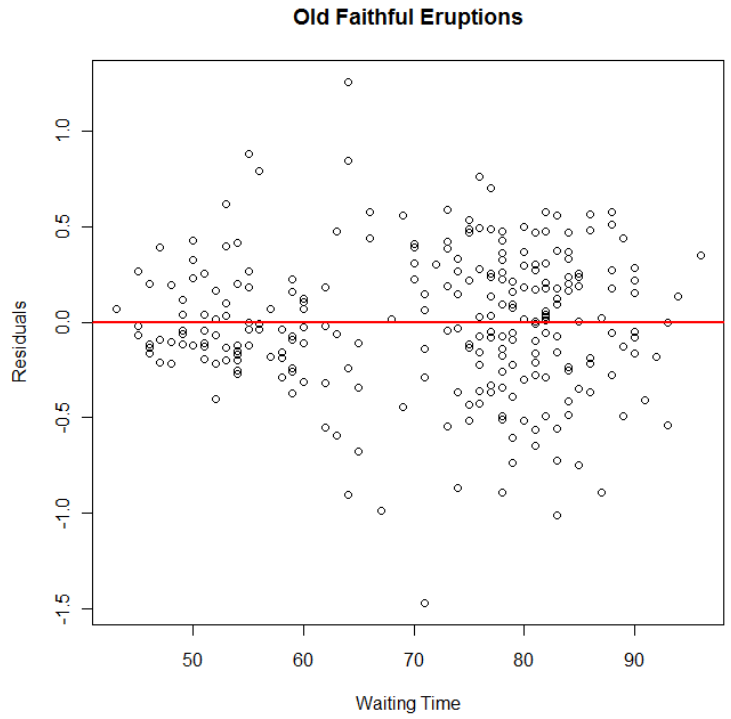



```
> summary(eruption.gam)

Family: gaussian
Link function: identity

Formula:
eruptions ~ 1 + s(waiting)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.48778    0.02242   155.6   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
             edf Ref.df     F p-value
s(waiting) 6.674  6.674 345.9  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.895   Deviance explained = 89.8%
GCV score = 0.14067  Scale est. = 0.1367    n = 272
```

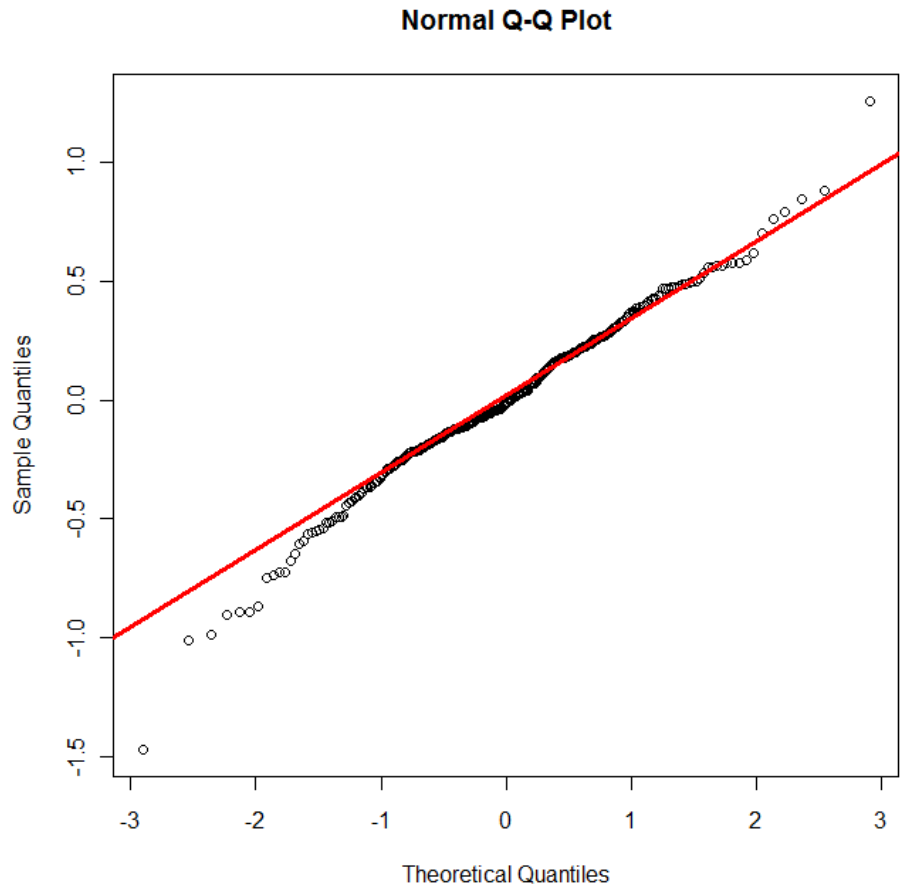
p-value

Coefficient of Determination

```
eruption.res.gam <- resid(eruption.gam)
plot(faithful$waiting, eruption.res.gam, ylab="Residuals",
        xlab="Waiting Time", main="Old Faithful Eruptions")
abline(0,0, col="red", lwd=2)
```



```
eruption.res.gam <- resid(eruption.gam)
qqnorm(eruption.res.gam)
qqline(eruption.res.gam, col="red", lwd=3)
```



Multiple regresion

- describes a dependent variable y (response) by independent variables x_1, x_2, ..., x_p ($p > 1$) is expressed by the equation

$$y = \alpha + \sum_k \beta_k x_k + \varepsilon$$

where the numbers α and β_k ($k = 1, 2, ..., p$) are the parameters, and ϵ is the error term.

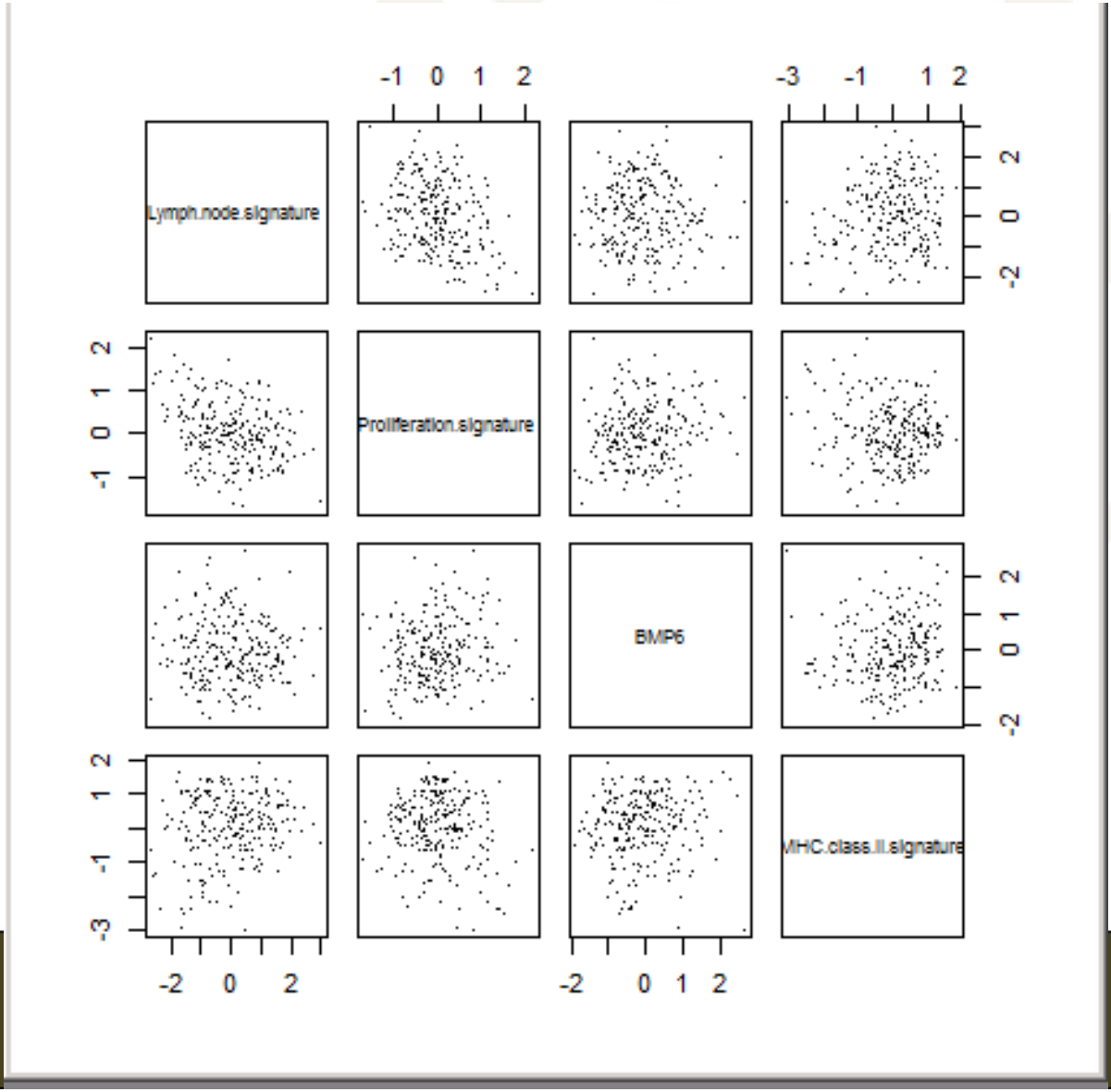
Multiple regresion
Explore your data

- Explore your data before starting the analysis
  - Are the responses correlated?
  - Use plot (trellis graphics, boxplot)
  - Use correlation (pearson, spearman)

DLBCL patient data

- Response: Germinal.cnter.B.cell.signature
- Explanatory variables
  - Lymph.node.signature
  - Proliferation.signature
  - BMP6
  - MHC.class.II.signature

pairs(dat[,8:11])



cor(dat[,8:11])

```
                         Lymph.node.signature Proliferation.signature
Lymph.node.signature               1.00000000               -0.3158644
Proliferation.signature           -0.31586437                1.0000000
BMP6                              -0.02659956                0.1407948
MHC.class.II.signature             0.15082317               -0.1341130
                                         BMP6 MHC.class.II.signature
Lymph.node.signature               -0.02659956                0.1508232
Proliferation.signature             0.14079484               -0.1341130
BMP6                                1.00000000                0.0865004
MHC.class.II.signature              0.08650040                1.0000000
```

```
> fit <-lm(Germinal.center.B.cell.signature ~ Lymph.node.signature +
+ Proliferation.signature + BMP6 + MHC.class.II.signature, data = dat)
> summary(fit)

Call:
lm(formula = Germinal.center.B.cell.signature ~ Lymph.node.signature +
    Proliferation.signature + BMP6 + MHC.class.II.signature,
    data = dat)

Residuals:
    Min      1Q  Median      3Q     Max
-2.5999 -0.7439  0.0084  0.8957  2.4159

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)
(Intercept)              -0.03625    0.07103  -0.510  0.61030
Lymph.node.signature      0.14283    0.06670   2.141  0.03327 *
Proliferation.signature   0.04374    0.11158   0.392  0.69539
BMP6                     -0.12930    0.08464  -1.528  0.12797
MHC.class.II.signature    0.19824    0.07589   2.612  0.00958 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.099 on 235 degrees of freedom
Multiple R-squared: 0.06031,    Adjusted R-squared: 0.04432
F-statistic: 3.771 on 4 and 235 DF,  p-value: 0.005413

> AIC(fit)
[1] 733.2417
```

p-vaules

Not significant p values

Adjusted Coefficient of Determination

Adjusted Coefficient of Determination

$$R^2_{adj} = 1 - (1 - R^2)\frac{n-1}{n-p-1}$$

Hvor n er antall observasjoner og
p er antall parametere brukt i modellen

Comparing models

- As long as analysis are done on the same responses you can compare your models by information criteria:
- AIC (Akaike's An Information Criterion)

  *-2*log-likelihood + 2*npar*
- *BIC* (Schwarz's Bayesian criterion)

  *-2*log-likelihood + log(n)*npar*
- Goal: as small AIC or BIC as possible, i.e. explain most with less parameters

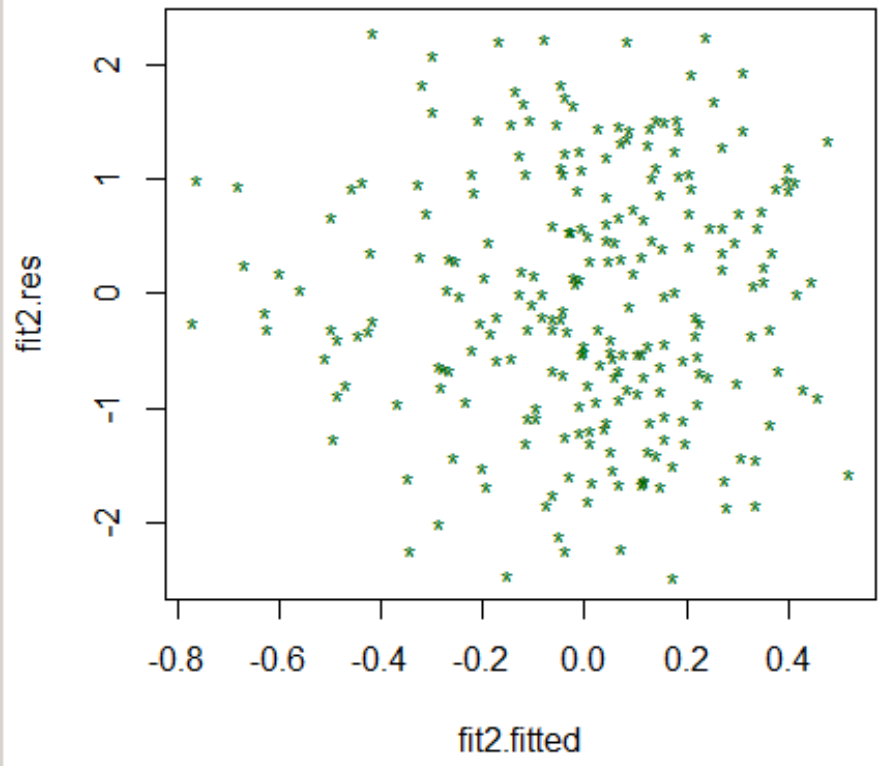
```
> fit2 <-lm(Germinal.center.B.cell.signature ~ -1 + Lymph.node.signature +
+  MHC.class.II.signature, data = dat)
> summary(fit2)

Call:
lm(formula = Germinal.center.B.cell.signature ~ -1 + Lymph.node.signature +
    MHC.class.II.signature, data = dat)

Residuals:
     Min       1Q   Median       3Q      Max
-2.47164 -0.82103 -0.09149  0.89951  2.28712

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
Lymph.node.signature    0.13862    0.06352   2.182   0.0301 *
MHC.class.II.signature  0.18504    0.07507   2.465   0.0144 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.098 on 238 degrees of freedom
Multiple R-squared: 0.05084,    Adjusted R-squared: 0.04287
F-statistic: 6.374 on 2 and 238 DF,  p-value: 0.00201

> AIC(fit2)
[1] 729.8265
>
>
```

All p values are significant

Adjusted coefficient of determination hardly changed

The AIC value is less, chose this model

```
fit2.res <- resid(fit2)
fit2.fitted <- fitted(fit2)
pairs(fit2.fitted, fit2.res, col = "darkgreen", pch="*")
```



Logistic regression

- We use the logistic regression equation to predict the probability of a dependent variable that takes on only the values 0 and 1. Suppose $x_1, x_2, \ldots, x_p$ ($p > 1$) are the independent variables, α, β_k ($k = 1, 2, \ldots, p$) are the parameters, and E(y) is the expected value of the dependent variable y

$$E(y) = \frac{1}{1 + e^{-\left(\alpha + \sum_k \beta_k x_k\right)}}$$

DLBCL patient data

- Response: alive or dead
- Explanatory variables
  - Subgroup
  - IPI.Group
  - Germinal.center.B.cell.signature
  - Lymph.node.signature
  - Proliferation.signature
  - BMP6
  - MHC.class.II.signature

DLBCL.glm <- glm(Status.at.follow.up ~ Subgroup + IPI.Group + Germinal.center.B.cell.signature + Lymph.node.signature + Proliferation.signature + BMP6 + MHC.class.II.signature, family= "binomial", data = dat)

Here logistic regression is chosen

```
> summary(DLBCL.glm)

Call:
glm(formula = Status.at.follow.up ~ Subgroup + IPI.Group + Germinal.center.B.cell.signat
    Lymph.node.signature + Proliferation.signature + BMP6 + MHC.class.II.signature,
    family = "binomial", data = DLBCL)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3745  -0.8724   0.4384   0.8099   2.3550

Coefficients:
                                Estimate Std. Error z value Pr(>|z|)
(Intercept)                       1.3144     0.5890   2.232   0.0256 *
SubgroupGCB                       0.5586     0.5386   1.037   0.2997
SubgroupType III                  0.8941     0.5635   1.587   0.1126
IPI.GroupLow                     -2.3313     0.5798  -4.021  5.8e-05 ***
IPI.GroupMedium                  -1.0145     0.5582  -1.818   0.0691 .
IPI.Groupmissing                -15.4343   882.7436  -0.017   0.9861
Germinal.center.B.cell.signature  -0.5073     0.2152  -2.358   0.0184 *
Lymph.node.signature              -0.1462     0.1560  -0.937   0.3486
Proliferation.signature            0.6619     0.3042   2.176   0.0295 *
BMP6                               0.4678     0.2068   2.262   0.0237 *
MHC.class.II.signature            -0.3844     0.1863  -2.063   0.0391 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 304.82  on 222  degrees of freedom
Residual deviance: 234.73  on 212  degrees of freedom
  (17 observations deleted due to missingness)
AIC: 256.73

Number of Fisher Scoring iterations: 13
```

DLBCL2.glm <- glm(Status.at.follow.up ~ IPI.Group, family= "binomial", data = dat)

summary(DLBCL2.glm)

High	Low	Medium	missing	NA's
32	82	108	1	17

```
> DLBCL2.glm <- glm(Status.at.follow.up ~ IPI.Group, family= "binomial", data = DLBCL)
> summary(DLBCL2.glm)

Call:
glm(formula = Status.at.follow.up ~ IPI.Group, family = "binomial",
    data = DLBCL)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.9268  -0.9140   0.5829   0.9005   1.4660

Coefficients:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)        1.6864     0.4869   3.464 0.000533 ***
IPI.GroupLow      -2.3432     0.5397  -4.342 1.41e-05 ***
IPI.GroupMedium   -0.9933     0.5279  -1.881 0.059913 .
IPI.Groupmissing -16.2525   882.7435  -0.018 0.985311
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 304.82  on 222  degrees of freedom
Residual deviance: 270.51  on 219  degrees of freedom
  (17 observations deleted due to missingness)
AIC: 278.51

Number of Fisher Scoring iterations: 13
```

The survival of those containing the IPI.group Low is significantly different from those that are in the group High

The estimate is negative, hence pations in group Low have less probability to die then those in group High

Backward an forward inclusion of response variables

- Forward
  - Start with all response variables, exclude the one that is least significant, compare AIC values.
- Backward
  - Start with one regression for each response variable
  - Include more and more response variables that are significant in the model, compare AIC values

Survival analysis

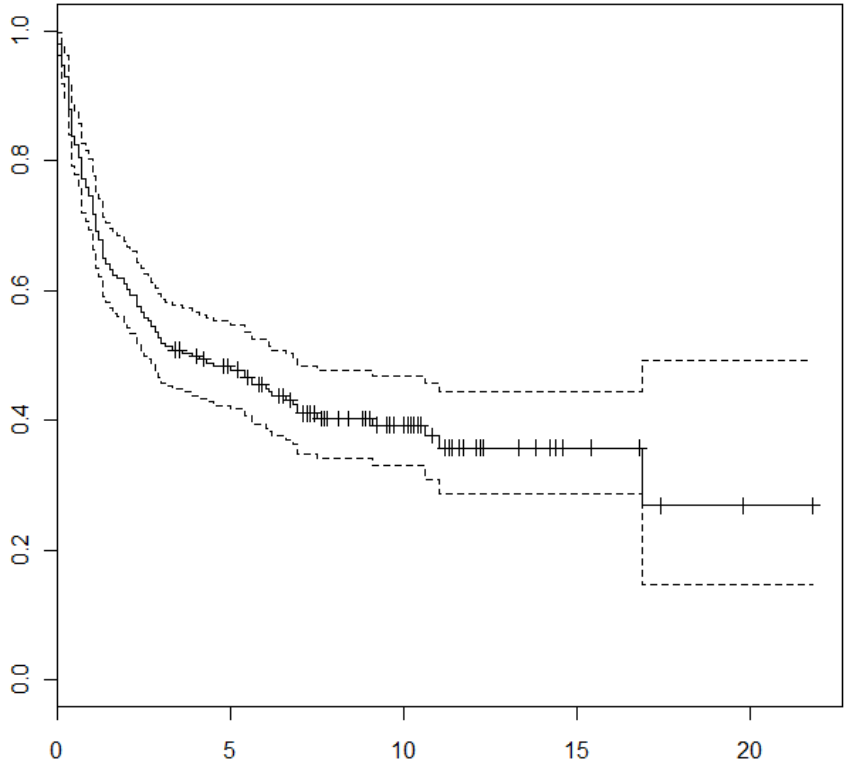
```
> install.packages("survival")
> library(survival)
Loading required package: splines
Attaching package: 'survival'
      The following object(s) are masked from
   package:boot :
       aml
```

Create a Survival Object

- Needs two arguments:
  - time: follow-up time
  - event: status indicator
- event=TRUE means event occurred
- event=FALSE indicates censoring
- Other values possible (see help(Surv))

```
surv.obj <- Surv(dat$Follow.up..years., dat$Status.at.follow.up == "Dead")
```

```
surv.survfit <- survfit(surv.obj~1)
plot(surv.survfit)
```

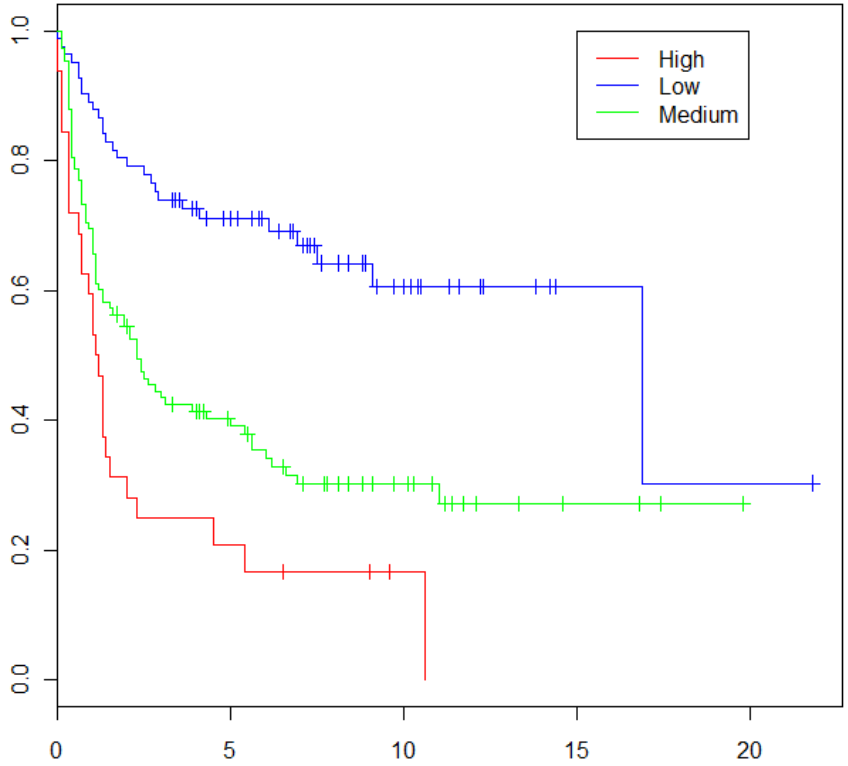


summary(surv.survfit)

Call: survfit(formula = surv.obj ~ 1)

time	n.risk	n.event	survival	std.err	lower 95% CI	upper 95% CI
0.0	240	5	0.979	0.00922	0.961	0.997
0.1	235	8	0.946	0.01461	0.918	0.975
0.2	227	4	0.929	0.01656	0.897	0.962
0.3	223	12	0.879	0.02104	0.839	0.921
0.4	211	10	0.837	0.02381	0.792	0.885
0.5	201	3	0.825	0.02453	0.778	0.874
0.6	198	5	0.804	0.02562	0.755	0.856
:						

```
surv.survfit.IPI <- survfit(Surv(Follow.up..years.,Status.at.follow.up=="Dead") ~
                IPI.Group, data = dat)

plot(surv.survfit.IPI, col = c("red", "blue", "green"))

legend(15,1,c("High", "Low", "Medium"), col = c("red", "blue", "green"), lty=c(1,1,1))
```



Test Survival Curve Differences

```
survdiff(Surv(Follow.up..years.,Status.at.follow.up=="Dead") ~ IPI.Group, data = dat)
Call:
survdiff(formula = Surv(Follow.up..years., Status.at.follow.up ==
    "Dead") ~ IPI.Group, data = dat)

n=222, 18 observations deleted due to missingness.
                      N  Observed    Expected        (O-E)^2/E        (O-E)^2/V
IPI.Group=High       32      27          12.4          17.33            19.93
IPI.Group=Low        82      28          57.5          15.11            28.60
IPI.Group=Medium    108      72          57.2           3.85             7.19

 Chisq= 37.8  on 2 degrees of freedom, p= 6.04e-09
```

Atleast two of the survival curves are different.