Curso de Estadística básica para Data Scientists

Dae-Jin Lee < lee.daejin@gmail.com >

TEMA 6. Modelos lineales

Índice

Regresar a la página principal

1. Modelos lineales en R

1.1. Regresión lineal simple

- La regresión es un método estadístico utilizado para predecir el valor de una variable de respuesta basada en los valores de un conjunto de variables explicativas.
- Una forma muy general para el modelo sería

$$y = f(x_1, x_2, ..., x_p) + \epsilon,$$

donde f es una función desconocida y ϵ es el error en esta representación. Dado que usualmente no tenemos suficientes datos para tratar de estimar f directamente (*problema inverso*), normalmente tenemos que suponer que tiene alguna forma restringida.

- Cualquier modelo estadístico intenta aproximar la variable de respuesta o variable dependiente y como una función matemática de las variables explicativas o regresores X (también llamadas covariables o variables independientes).
- La forma más sencilla y más común es la **regresión lineal**

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_z + \epsilon,$$

donde β_i $i = 0, 1, 2$ son parametros *desconocidos*. β_0 se llama el intercepto. Por lo tanto, el problema se reduce a la estimación de cuatro valores en lugar de la complicada infinita dimensión f.

- Un modelo lineal simple con una sola variable explicativa se define como:

$$\hat{y} = \beta_0 + \beta_1 x$$

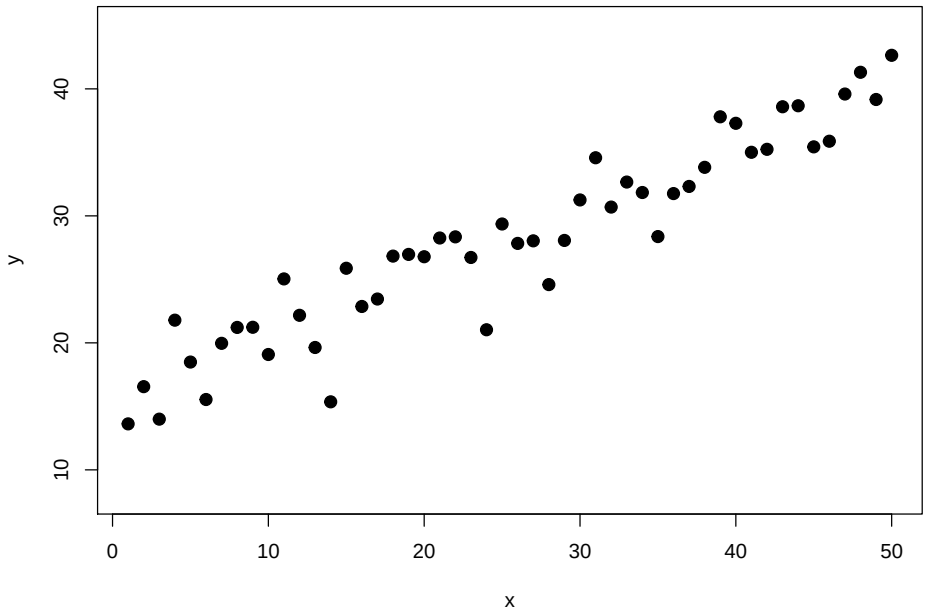
donde $\hat{y}$ son los valores ajustados para β_0 (intercepto) y β_1 (pendiente). Entonces para un x_i dado obtenemos un $\hat{y}_i$ que se aproxima a y_i

Supongamos el siguiente ejemplo simulado (con $p = 1$):

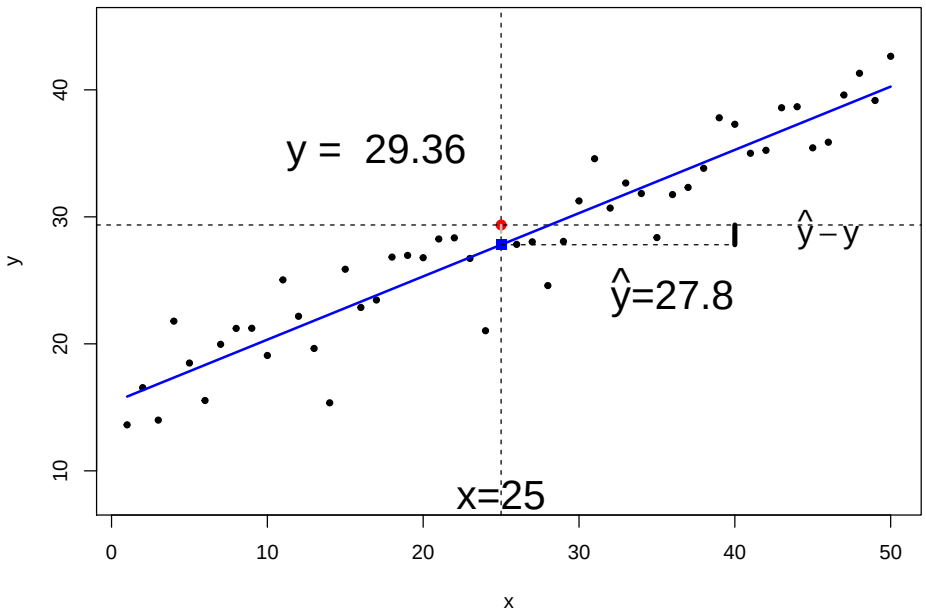
```r
set.seed(1)
n <- 50

x <- seq(1,n)
 beta0 <- 15
 beta1 <- 0.5

sigma <- 3 # standar deviation of the errors
eps <- rnorm(n,mean=0,sd=3) # generate gaussian random errors

# Generate random data
 y <- beta0 + beta1*x  +  eps
```

Plot de los datos

```r
plot(x,y,ylim = c(8,45), cex=1.3, xlab = "x", ylab="y",pch=19)
```

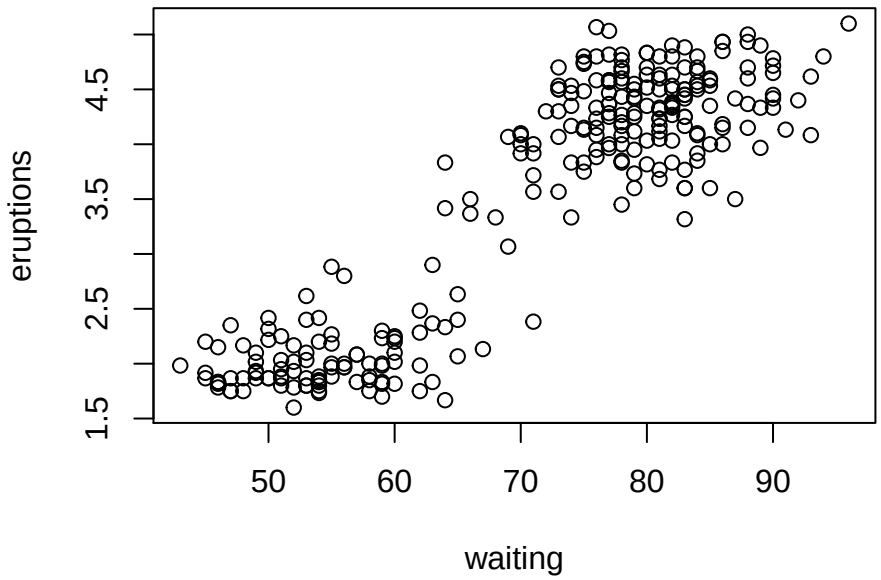


Procedimiento matemático para encontrar la curva que mejor se ajuste a un conjunto de puntos dado, resulta de minimizar la suma de los cuadrados de los residuos de los puntos de la línea ajustada. Ilustración del ajuste de mínimos cuadrados



Por ejemplo, en el conjunto de datos `faithful`, contiene datos de ejemplo de dos variables aleatorias denominadas `waiting` y `eruptions`. La variable `waiting` indica el tiempo de espera hasta las próximas erupciones, y`eruptions` denota la duración.

```
plot(eruptions~waiting,data=faithful)
```



El modelo lineal viene dado por:

$$Eruptions = \beta_0 + \beta_1 * Waiting + \epsilon$$

Si seleccionamos los parámetros β_0 y β_1 en el modelo de regresión lineal simple para minimizar la suma de cuadrados del término de error ϵ. Supongamos que para el conjunto de datos `faithful`, pretendemos estimar la siguiente duración de la erupción si el tiempo de espera desde la última erupción ha sido de 80 minutos. Aplicamos la función `lm` a una fórmula que describe la variable erupciones por la variable waiting, y guardamos el modelo de regresión lineal en una nueva variable `eruption.lm`.

```
data("faithful")
eruption.lm <- lm(eruptions~waiting,data=faithful)
```

Extraemos los parámetros de la ecuación de regresión estimada con la función de coeficientes.

```
coeffs <- coefficients(eruption.lm); coeffs
```

```
## (Intercept)     waiting
## -1.87401599  0.07562795
```

Ahora ajustamos la duración de la erupción usando la ecuación de regresión estimada.

```
waiting = 80           # the waiting time
duration = coeffs[1] + coeffs[2]*waiting
duration
```

```
## (Intercept)
##     4.17622
```

En base al modelo de regresión lineal simple, si el tiempo de espera desde la última erupción ha sido de 80 minutos, esperamos que los próximos minutos de duración 4.1762198.

Podemos incluir el valor de `waiting=80` en `newdata` como un `data.frame`

```
newdata = data.frame(waiting=80) # wrap the parameter
```

Y aplicar la función `predict` a modelo `eruption.lm` con `newdata`.

```r
predict(eruption.lm, newdata)    # apply predict
```

```
##       1
## 4.17622
```

Podemos calcular directamente las cantidades de interés, es decir, la solución de mínimos cuadrados ordinarios consiste en:

$$\min_{\beta_0,\beta_1} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Por tanto $\hat{\beta}_1 = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2}$ and $\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$

En forma de matricial, con $X = [1 : x_1 : ... : x_p]$

$$\hat{\beta} = (X'X)^{-1}X'$$

donde $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1)$

1.2. Definir modelos en R

Para completar una regresión lineal usando R es necesario primero entender la sintaxis para definir modelos.

Syntax	Model	Comments
`y ~ x`	$y = \beta_0 + \beta_1 x$	Linea recta con intercepto
`y ~ -1 + x`	$y = \beta_1 x$	Linea recta sin intercepto, i.e. la recta pasa por (0,0)
`y ~ x + I(x^2)`	$y = \beta_0 + \beta_1 x + \beta_2 x^2$	Modelo polinomial; `I()` permite incluir símbolos matemáticos
`y ~ x + z`	$y = \beta_0 + \beta_1 x + \beta_2 z$	Model de regresión múltiple
`y ~ x:z`	$y = \beta_0 + \beta_1 xz$	Modelo con interacción entre x y z
`y ~ x*z`	$y = \beta_0 + \beta_1 x + \beta_2 z + \beta_3 xz$	Equivale a `y~x+z+x:z`

En R la función `model.matrix` permite crear la matriz de diseño X.

```r
x <- model.matrix(~waiting, data = faithful)
y <- faithful$eruptions
xtxi <- solve(t(x) %*% x)
```

```
betas <- xtxi %*% t(x) %*% y
betas
```

```
##                     [,1]
## (Intercept) -1.87401599
## waiting      0.07562795
```

o

```
solve(crossprod(x, x), crossprod(x, y))
```

```
##                     [,1]
## (Intercept) -1.87401599
## waiting      0.07562795
```

La función `lm()` permite calcular internamente el modelo lineal de regresión

Se puede obtener $(X'X)^{-1}$ como

```
summary(eruption.lm)$cov.unscaled
```

```
##              (Intercept)       waiting
## (Intercept)  0.104029479 -1.415475e-03
## waiting     -0.001415475  1.996521e-05
```

Con el comando `names()` podemos ver los componentes de un objeto de `R`

```
names(eruption.lm)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

Por ejemplo:

- Valores ajustados (o predichos) ($\hat{y}$):

```
eruption.lm$fitted.values
```

- Residuos ($y - \hat{y}$)

```
eruption.lm$residuals
```

Podemos estimar σ como $\sigma = \frac{(y_i - \hat{y}_i)^2}{n-p}$ en R

```
eruption.lm.sum <- summary(eruption.lm)
names(eruption.lm.sum)
```

```
##  [1] "call"          "terms"         "residuals"     "coefficients"
##  [5] "aliased"       "sigma"         "df"            "r.squared"
##  [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
```

```
sqrt(deviance(eruption.lm)/df.residual(eruption.lm))
```

```
## [1] 0.4965129
```

```
# is obtained directly as
eruption.lm.sum$sigma
```

```
## [1] 0.4965129
```

También podemos obtener los errores estándar para los coeficientes. También `diag()` devuelve la diagonal de un matriz:

```
xtxi
```

```
##              (Intercept)       waiting
## (Intercept)  0.104029479 -1.415475e-03
## waiting     -0.001415475  1.996521e-05
```

```
sqrt(diag(xtxi)) * eruption.lm.sum$sigma
```

```
## (Intercept)     waiting
## 0.160143302 0.002218541
```

```
eruption.lm.sum$coef[, 2]
```

```
## (Intercept)     waiting
## 0.160143302 0.002218541
```

1.3. Coeficiente de determinación

El **coeficiente de determinación** de un modelo de regresión lineal es el cociente de las varianzas de los valores ajustados y los valores observados de la variable dependiente. Si denotamos y_i como los valores observados de la variable dependiente, $\bar{y}$ como su media, y $\bar{y}_i$ como el valor ajustado, entonces el coeficiente de determinación es:

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

```
summary(eruption.lm)$r.squared
```

```
## [1] 0.8114608
```

o

```
1-sum(eruption.lm$res^2)/sum((y-mean(y))^2)
```

```
## [1] 0.8114608
```

Más opciones:

- `fitted.values()` o `fitted()` valores ajustados
- `predict()`: valores predichos $\hat{y}_*$ para valores de x_*
- `confint()`: intervalos de confianza para los parámetros del modelo
- `resid()`: residuos del modelo
- `anova()`: Tabla de analisis de la varianza para los residuos
- `deviance()`: devianza del modelo ajustado, en el caso de la regresión lineal $\sum_i^n(\hat{y}_i - y_i)^2$

*Ver libro de Faraway (2002) book (Chapters 1-7)*aquí

1.4. Prueba de significancia para la regresión lineal

Supongamos que el término de error en el modelo de regresión lineal es independiente de x, y se distribuye normalmente, con media cero y varianza constante. Podemos decidir si existe alguna relación significativa entre x e y probando la hipótesis nula de que $\beta_1 = 0$.

El resultado del test es el estadístico $F $

```r
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
```

1.5. Intervalo de confianza para la regresión lineal

Supongamos que el término de error ϵ en el modelo de regresión lineal es independiente de x, y se distribuye normalmente, con media cero y varianza constante. Para un valor dado de x, la estimación de intervalo para la media de la variable dependiente, $\bar{y}$, se llama intervalo de confianza.

Un intervalo de confianza del 95 % de la duración media de la erupción para el tiempo de espera de 80 minutos está dado por

```r
predict(eruption.lm, newdata, interval="confidence")
```

```
##       fit      lwr      upr
## 1 4.17622 4.104848 4.247592
```

El intervalo de confianza del 95 % de la duración media de la erupción para el tiempo de espera de 80 minutos está entre 4.1048 y 4.2476 minutos.

1.6. Intervalo de predicción para la regresión lineal

Para un valor dado de x, la estimación de intervalo de la variable dependiente y se denomina intervalo de predicción.

```
predict(eruption.lm, newdata, interval="predict")
```

```
##       fit      lwr      upr
## 1 4.17622 3.196089 5.156351
```

El intervalo de predicción del 95 % de la duración de la erupción para el tiempo de espera de 80 minutos está entre 3.1961 y 5.1564 minutos.

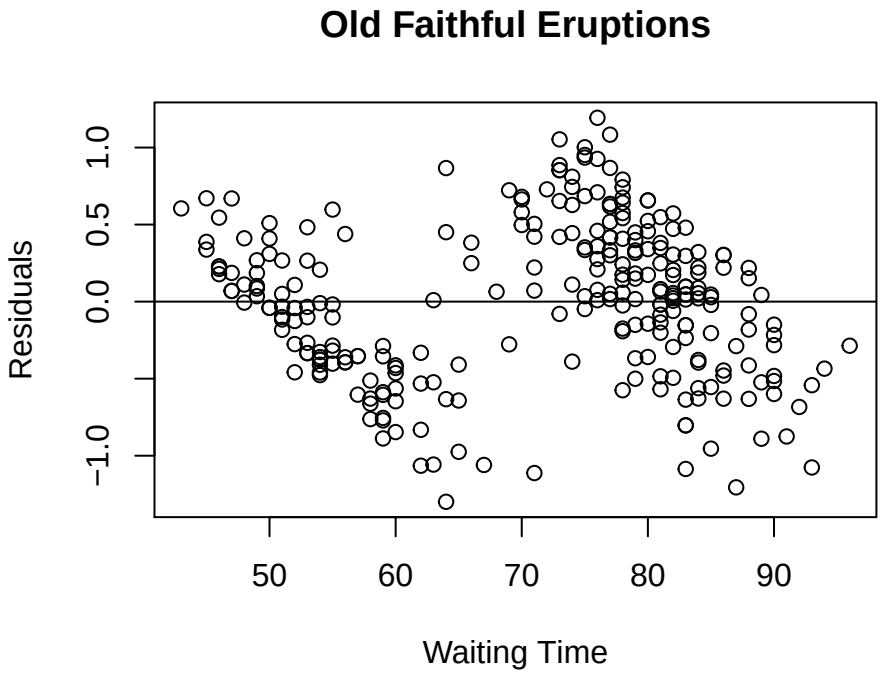
1.7. Residual Plot

Los residuos del modelo de regresión lineal simple son la diferencia entre los datos observados de la variable dependiente y y los valores ajustados $ hat{y}$.

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

```
eruption.res = resid(eruption.lm)
```

```
plot(faithful$waiting,
    eruption.res,
    ylab="Residuals", xlab="Waiting Time",
    main="Old Faithful Eruptions")
abline(0, 0)
```



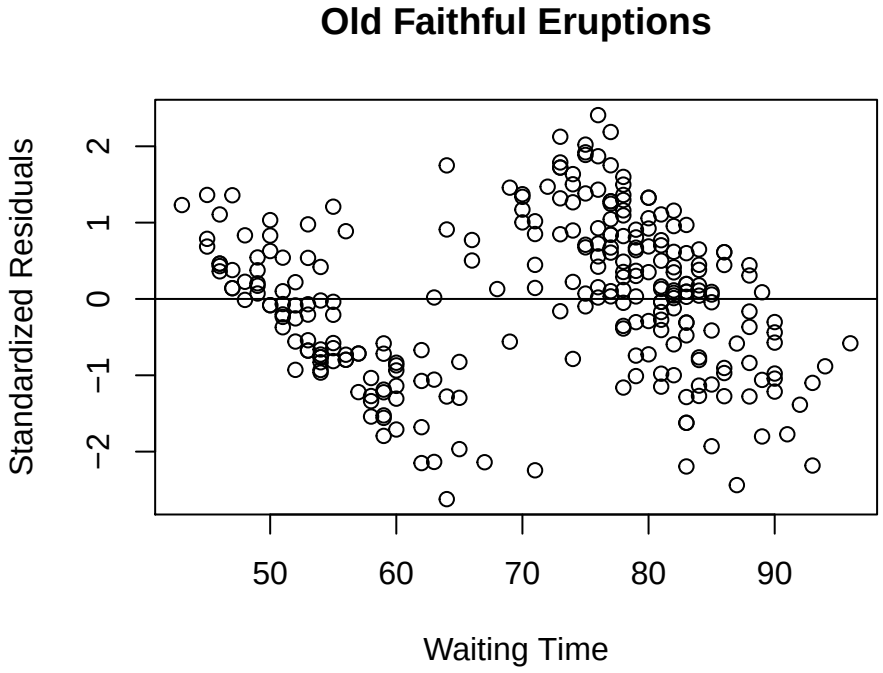
1.8. Residuo estandarizado

El residuo estandarizado es el residuo dividido por su desviación estándar.

$$\text{Standardized residual}_i = \frac{Residual_i}{SD.of.Residual_i}$$

```
eruption.stdres <- rstandard(eruption.lm)
```

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



Como el p-valor es mucho menor que 0,05, rechazamos la hipótesis nula de que $\beta_1 = 0$. Por lo tanto, existe una relación significativa entre las variables en el modelo de regresión lineal del conjunto de datos `faithful`.

1.9. Gráfico de normalidad de los residuos

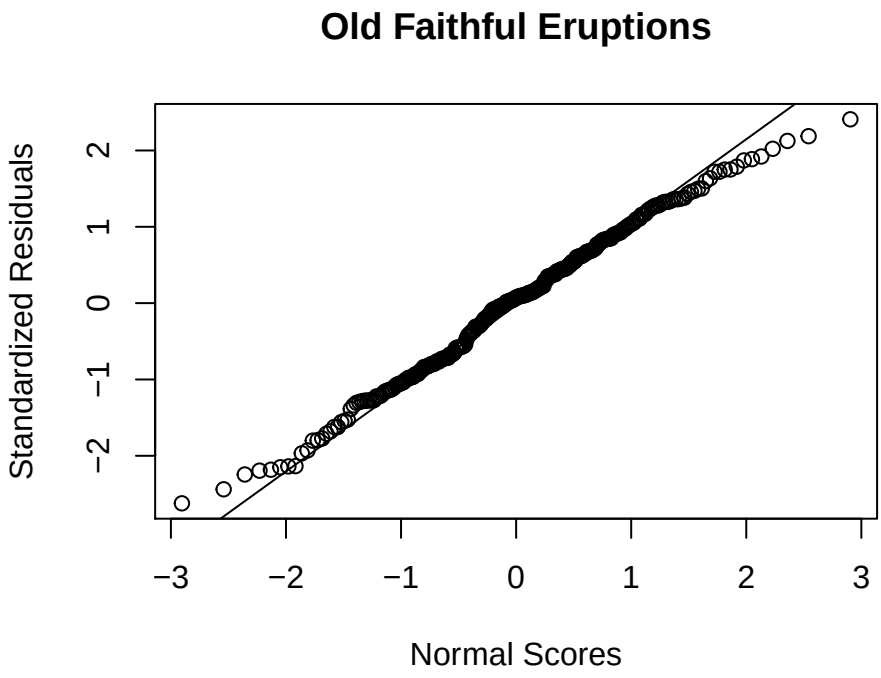
El gráfico de probabilidad normal es una herramienta gráfica para comparar un conjunto de datos con la distribución normal. Podemos usarlo con el residuo

estandarizado del modelo de regresión lineal y ver si el término de error ϵ es realmente distribuido normalmente.

```
eruption.lm = lm(eruptions ~ waiting, data=faithful)
eruption.stdres = rstandard(eruption.lm)
```

Ahora creamos el gráfico de probabilidad normal con la función `qqnorm` y añadimos`qqline` para una comparación posterior.

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



1.10. Regresión lineal múltiple

Un modelo de regresión lineal múltiple describe una variable dependiente y por variables independientes $x_1, x_2, ..., x_p$ $(p > 1)$ se expresa mediante la ecuación:

$$y = \beta_0 + \sum_{k}^{p} \beta_k + \epsilon$$

donde β_0 y β_k $(k = 1, 2, ..., p)$ son los parámetros y ϵ el término de error.

Example:

Consideremos los datos `stackloss`, sea `stackloss` la variable dependiente, y`Air.Flow` (cooling air flow), `Water.Temp` (inlet water temperature) y `Acid.Conc.` (acid concentration) las variables independientes, la regresión multiple es:

$$stack.loss = \beta_0 + \beta_1 * Air.Flow + \beta_2 * Water.Temp + \beta_3 * Acid.Conc + \epsilon$$

```
data("stackloss")
?stackloss
head(stackloss)
```

```
##   Air.Flow Water.Temp Acid.Conc. stack.loss
## 1       80         27         89         42
## 2       80         27         88         37
## 3       75         25         90         37
## 4       62         24         87         28
## 5       62         22         87         18
## 6       62         23         87         18
```

El modelo de regresión multiple es:

```
stackloss.lm = lm(stack.loss ~ Air.Flow + Water.Temp + Acid.Conc., data=stackloss)
stackloss.lm
```

```
##
## Call:
## lm(formula = stack.loss ~ Air.Flow + Water.Temp + Acid.Conc.,
##     data = stackloss)
##
## Coefficients:
## (Intercept)     Air.Flow   Water.Temp   Acid.Conc.
##    -39.9197       0.7156       1.2953      -0.1521
```

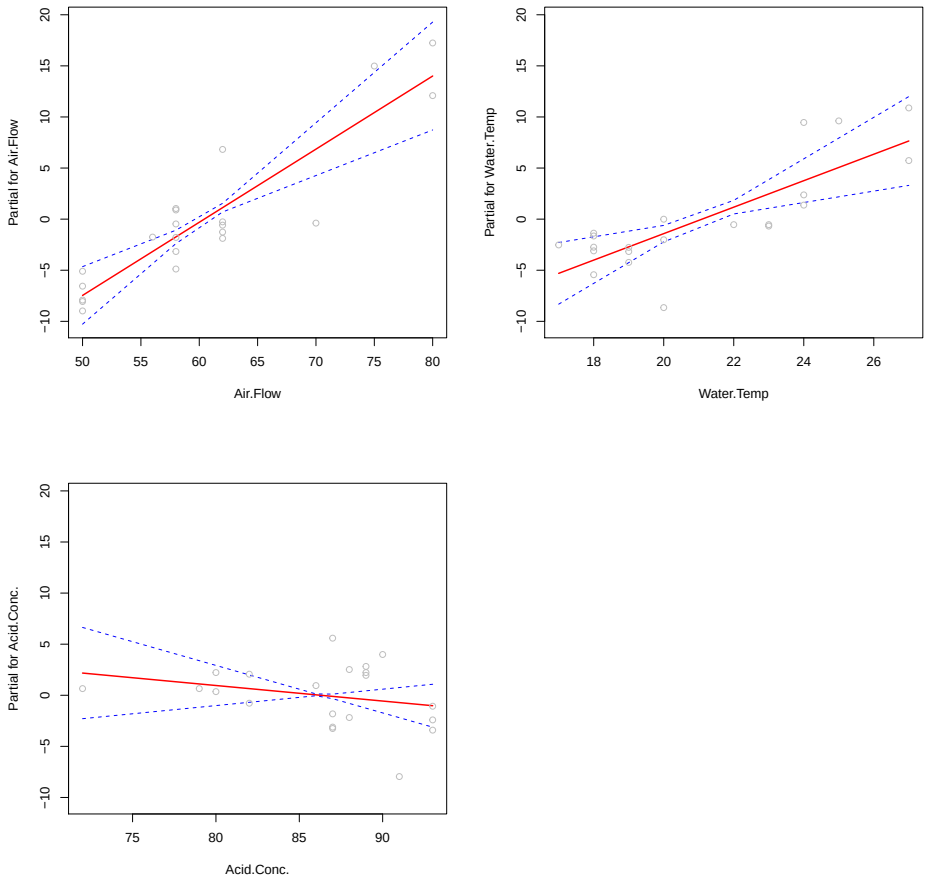
```
summary(stackloss.lm)
```

```
##
## Call:
## lm(formula = stack.loss ~ Air.Flow + Water.Temp + Acid.Conc.,
##     data = stackloss)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.2377 -1.7117 -0.4551  2.3614  5.6978
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -39.9197    11.8960  -3.356  0.00375 **
## Air.Flow      0.7156     0.1349   5.307  5.8e-05 ***
## Water.Temp    1.2953     0.3680   3.520  0.00263 **
## Acid.Conc.   -0.1521     0.1563  -0.973  0.34405
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.243 on 17 degrees of freedom
## Multiple R-squared:  0.9136, Adjusted R-squared:  0.8983
## F-statistic:  59.9 on 3 and 17 DF,  p-value: 3.016e-09
```

La función `termplot` permite representar los términos de la regresión frente a las variables predictoras:

```
?termplot
par(mfrow=c(2,2))
termplot(stackloss.lm, partial.resid = TRUE, se=TRUE,col.se = "blue")
```



What is the stack loss if the air flow is 72, water temperature is 20 and acid concentration is 85?

Crear un nuevo `data.frame`:

```
newdata <- data.frame(Air.Flow=72,Water.Temp=20,Acid.Conc.=85)
```

Con `predict`

```
predict(stackloss.lm, newdata)
```

```
##        1
## 24.58173
```

Basado en el modelo de regresión lineal múltiple y los parámetros dados, la pérdida prevista es 24.5817284.

Para obtener el coeficiente de determinación múltiple

```r
summary(stackloss.lm)$r.squared
```

```
## [1] 0.9135769
```

1.10.1. Coeficiente de determinación ajustado

El coeficiente de determinación ajustado de un modelo de regresión lineal múltiple se define en términos del coeficiente de determinación como sigue, donde n es el número de observaciones en el conjunto de datos y p es el número de variables independientes.

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

```r
summary(stackloss.lm)$adj.r.squared
```

```
## [1] 0.8983258
```

1.10.2. Pruebas de significación e intervalos de confianza / predicción

```r
summary(stackloss.lm)
```

```
##
## Call:
## lm(formula = stack.loss ~ Air.Flow + Water.Temp + Acid.Conc.,
##     data = stackloss)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.2377 -1.7117 -0.4551  2.3614  5.6978
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -39.9197    11.8960  -3.356  0.00375 **
## Air.Flow      0.7156     0.1349   5.307  5.8e-05 ***
## Water.Temp    1.2953     0.3680   3.520  0.00263 **
## Acid.Conc.   -0.1521     0.1563  -0.973  0.34405
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 3.243 on 17 degrees of freedom
## Multiple R-squared:  0.9136, Adjusted R-squared:  0.8983
## F-statistic:  59.9 on 3 and 17 DF,  p-value: 3.016e-09
```

Como los p-valores de `Air.Flow` y`Water.Temp` son inferiores a 0,05, ambos son estadísticamente significativos en el modelo de regresión lineal múltiple de `stackloss`.

El intervalo de confianza al 95 % de `stackloss` es

```
predict(stackloss.lm, newdata, interval="confidence")
```

```
##        fit      lwr    upr
## 1 24.58173 20.21846 28.945
```

Y el invervalo de predicción al 95 %

```
predict(stackloss.lm, newdata, interval="prediction")
```

```
##        fit     lwr      upr
## 1 24.58173 16.4661 32.69736
```

1.11. Regresión lineal con variables factor

Supongamos el conjunto de datos `mtcars`.

```
data(mtcars)
t.test(mpg ~ am, data=mtcars)
```

```
##
##  Welch Two Sample t-test
##
## data:  mpg by am
## t = -3.7671, df = 18.332, p-value = 0.001374
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -11.280194  -3.209684
## sample estimates:
## mean in group 0 mean in group 1
##        17.14737        24.39231
```

Los resultados de las pruebas estadísticas se centran en `mpg` y`am` solamente, sin controlar las influencias de otras variables.

```
fit0 <- lm(mpg ~ factor(am), data = mtcars)
summary(fit0)
```

```
##
## Call:
## lm(formula = mpg ~ factor(am), data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.3923 -3.0923 -0.2974  3.2439  9.5077
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   17.147      1.125  15.247 1.13e-15 ***
## factor(am)1    7.245      1.764   4.106 0.000285 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.902 on 30 degrees of freedom
## Multiple R-squared:  0.3598, Adjusted R-squared:  0.3385
## F-statistic: 16.86 on 1 and 30 DF,  p-value: 0.000285
```

Si aplicamos una regresión multiple para controlar ciertas variables disponibles de diseño y rendimiento, el impacto marginal de los automóviles de transmisión automática o manual no resulta significativo. Las variables de confusión incluyen desplazamiento (`disp`), relación del eje trasero (`drat`) y peso del coche (`wt`). Supongamos el peso del coche (`wt`) por ejemplo. La regresión sugiere que, manteniendo constantes otras variables (ceteris paribus), los automóviles traídos por tracción consumen -0.024 galones más de gas por milla y los resultados ya no son estadísticamente significativos. Un análisis similar se puede observar para las otras dos variables: `drat` y `wt`.

```
fit1 <- lm(mpg ~ factor(am) + wt, data = mtcars)
summary(fit1)
```

```
##
## Call:
## lm(formula = mpg ~ factor(am) + wt, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5295 -2.3619 -0.1317  1.4025  6.8782
##
## Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 37.32155    3.05464  12.218 5.84e-13 ***
## factor(am)1 -0.02362    1.54565  -0.015    0.988
## wt          -5.35281    0.78824  -6.791 1.87e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.098 on 29 degrees of freedom
## Multiple R-squared:  0.7528, Adjusted R-squared:  0.7358
## F-statistic: 44.17 on 2 and 29 DF,  p-value: 1.579e-09
```

```
fit2 <- lm(mpg ~ factor(am) + drat, data = mtcars)
summary(fit2)
```

```
##
## Call:
## lm(formula = mpg ~ factor(am) + drat, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.5802 -2.5206 -0.5153  2.4419  8.5198
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -1.950      7.073  -0.276   0.7848
## factor(am)1    2.807      2.282   1.230   0.2286
## drat           5.811      2.130   2.728   0.0107 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.448 on 29 degrees of freedom
## Multiple R-squared:  0.4906, Adjusted R-squared:  0.4554
## F-statistic: 13.96 on 2 and 29 DF,  p-value: 5.659e-05
```

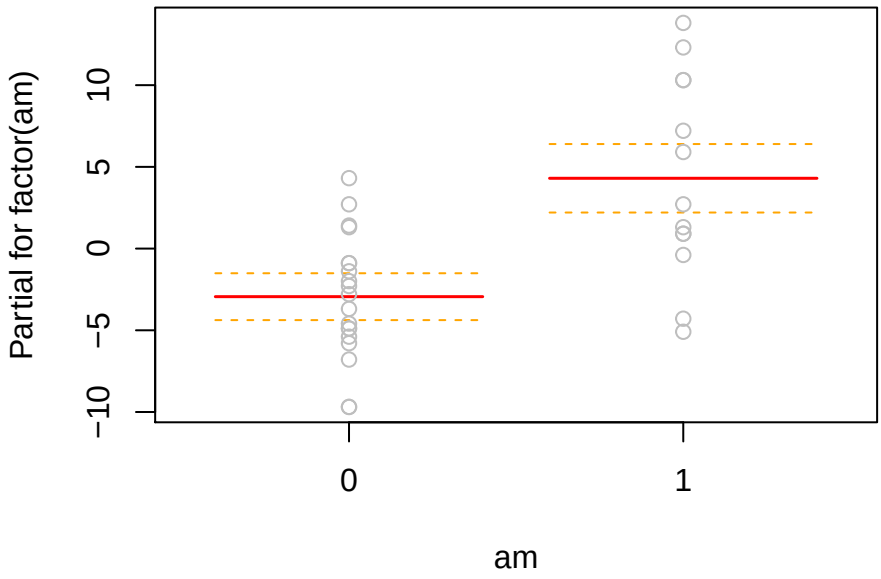
```
fit3 <- lm(mpg ~ factor(am) + disp, data = mtcars)
summary(fit3)
```

```
##
## Call:
## lm(formula = mpg ~ factor(am) + disp, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6382 -2.4751 -0.5631  2.2333  6.8386
```

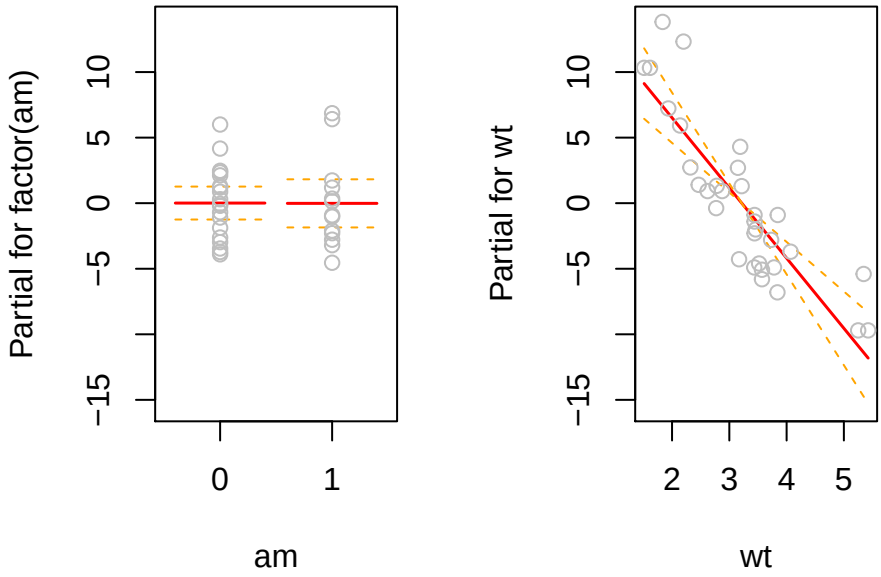
```
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 27.848081   1.834071  15.184 2.45e-15 ***
## factor(am)1  1.833458   1.436100   1.277    0.212
## disp        -0.036851   0.005782  -6.373 5.75e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.218 on 29 degrees of freedom
## Multiple R-squared:  0.7333, Adjusted R-squared:  0.7149
## F-statistic: 39.87 on 2 and 29 DF,  p-value: 4.749e-09
```

Con `termplot`

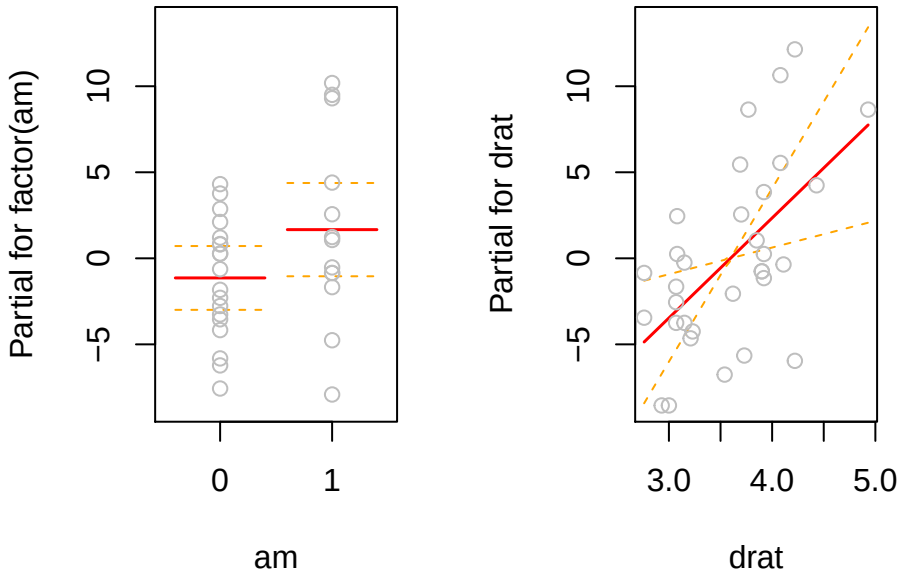
```
termplot(fit0,partial.resid = TRUE,se=TRUE)
```




```
par(mfrow=c(1,2))
termplot(fit1,partial.resid = TRUE,se=TRUE)
```




```
termplot(fit2,partial.resid = TRUE,se=TRUE)
```



1.12. Inferencia de modelos lineales

Comparación de modelos: test de razón de verosimitudes

Es una prueba estadística utilizada para comparar la bondad de ajuste de dos modelos, uno de los cuales (el modelo nulo) es un caso especial del otro (el

modelo alternativo). La prueba se basa en la razón de verosimilitud, que expresa cuántas veces más probabilidades de que los datos estén bajo un modelo que el otro.

La comparación de dos modelos ajustados a los mismos datos se puede configurar como un problema de prueba de hipótesis. Sea M_0 y M_1 los modelos.

Consideremos como la hipótesis nula *"$ M_1 $ no es una mejora significativa en $ M_0 $"*, y la alternativa la negación. Esta hipótesis se puede formular a menudo de modo que un estadístico se pueda generar de los dos modelos.

Normalmente, los modelos se anidan en que las variables en M_0 son un subconjunto de los de M_1. La estadística a menudo involucra los valores RSS (suma residual de cuadrados) para ambos modelos, ajustados por el número de parámetros utilizados. En la regresión lineal esto se convierte en una prueba de `anova` (comparando las varianzas).

```
M0 <- lm(mpg ~ 1, data = mtcars)
M1 <- lm(mpg ~ factor(am), data = mtcars)
summary(M0)
```

```
##
## Call:
## lm(formula = mpg ~ 1, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.6906 -4.6656 -0.8906  2.7094 13.8094
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   20.091      1.065   18.86   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.027 on 31 degrees of freedom
```

```
summary(M1)
```

```
##
## Call:
## lm(formula = mpg ~ factor(am), data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.3923 -3.0923 -0.2974  3.2439  9.5077
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   17.147      1.125  15.247 1.13e-15 ***
## factor(am)1    7.245      1.764   4.106 0.000285 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.902 on 30 degrees of freedom
## Multiple R-squared:  0.3598, Adjusted R-squared:  0.3385
## F-statistic: 16.86 on 1 and 30 DF,  p-value: 0.000285
```

```
anova(M0,M1)
```

```
## Analysis of Variance Table
##
## Model 1: mpg ~ 1
## Model 2: mpg ~ factor(am)
##   Res.Df    RSS Df Sum of Sq     F   Pr(>F)
## 1     31 1126.0
## 2     30  720.9  1    405.15 16.86 0.000285 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

La prueba/test de razón de verosimilitud también puede probar la significación de los predictores. Por lo tanto, podemos comparar el modelo `fit0` (donde`am` es significativo) con `fit1`,`fit2` o `fit3`, es decir:

```
anova(fit0, fit1)
```

```
## Analysis of Variance Table
##
## Model 1: mpg ~ factor(am)
## Model 2: mpg ~ factor(am) + wt
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1     30 720.90
## 2     29 278.32  1    442.58 46.115 1.867e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(fit0, fit2)
```

```
## Analysis of Variance Table
```

```
##
## Model 1: mpg ~ factor(am)
## Model 2: mpg ~ factor(am) + drat
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     30 720.90
## 2     29 573.64  1    147.26 7.4444 0.0107 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(fit0, fit3)
```

```
## Analysis of Variance Table
##
## Model 1: mpg ~ factor(am)
## Model 2: mpg ~ factor(am) + disp
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1     30 720.90
## 2     29 300.28  1    420.62 40.621 5.748e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Sin embargo, las pruebas de razón de verosimilitud sugieren que es importante considerar estas dimensiones (es decir, el desplazamiento, la relación del eje trasero y el peso) ya que estas variables aumentan el ajuste del modelo.

```
library(faraway)
data(pima, package="faraway")
head(pima)
summary(pima)
sort(pima$diastolic)
pima$diastolic[pima$diastolic == 0]  <- NA
pima$glucose[pima$glucose == 0] <- NA
pima$triceps[pima$triceps == 0]  <- NA
pima$insulin[pima$insulin == 0] <- NA
pima$bmi[pima$bmi == 0] <- NA
pima$test <- factor(pima$test)
summary(pima$test)
levels(pima$test) <- c("negative","positive")
summary(pima)
hist(pima$diastolic,xlab="Diastolic",main="")
plot(density(pima$diastolic,na.rm=TRUE),main="")
plot(sort(pima$diastolic),ylab="Sorted Diastolic")
plot(diabetes ~ diastolic,pima)
plot(diabetes ~ test,pima)
require(ggplot2)
```

```
ggplot(pima,aes(x=diastolic))+geom_histogram()
ggplot(pima,aes(x=diastolic))+geom_density()
ggplot(pima,aes(x=diastolic,y=diabetes))+geom_point()
ggplot(pima,aes(x=diastolic,y=diabetes,shape=test))+geom_point()+theme(legend.position = "to
ggplot(pima,aes(x=diastolic,y=diabetes)) + geom_point(size=1) + facet_grid(~ test)
```