Linear Regression

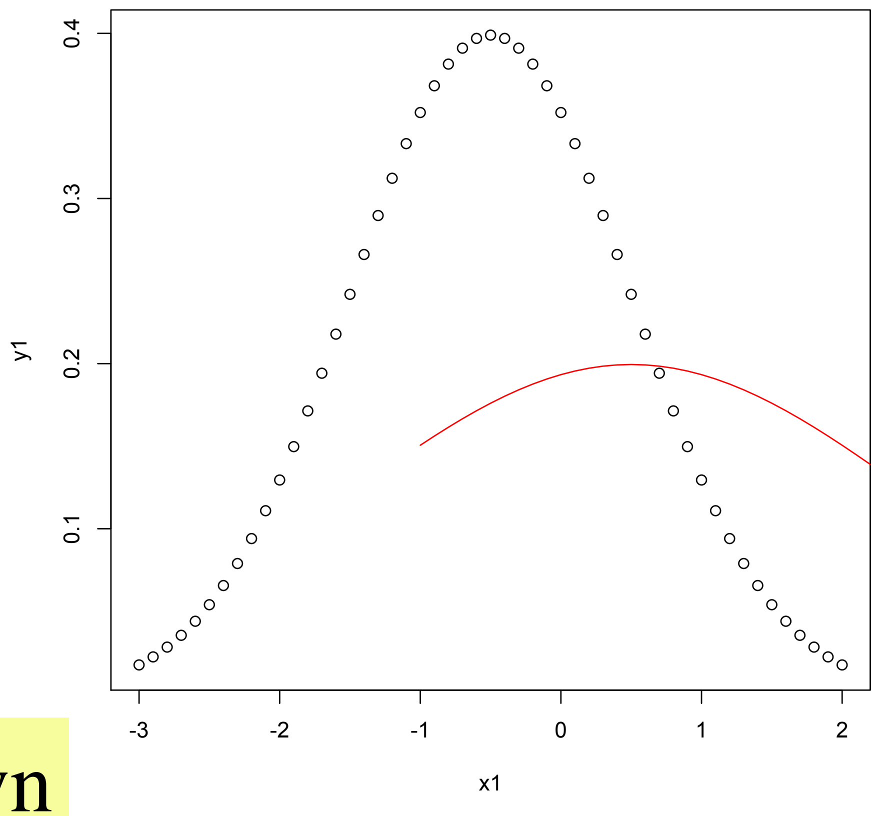
A few Problems

Plotting two datasets

- Often, one wishes to overlay two line plots over each other
- The range of the x and y variables might not be the same

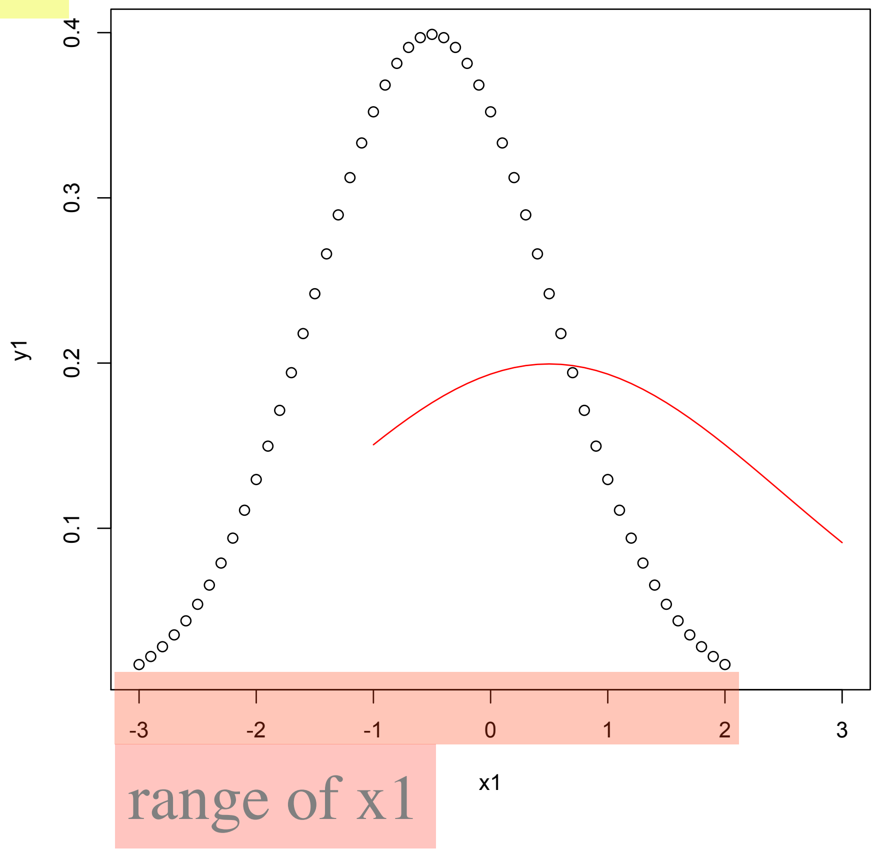
```
> x1 = seq(-3,2,.1)
> x2 = seq(-1,3,.1)
> y1 = dnorm(x1,mean=-.5)
> y2 = dnorm(x2,mean=.5,sd=2)

plot(x1,y1)
points(x2,y2,type='l',col='red')
```

Only the x values from x1 are shown



```
> xrg = range(x1,x2)
> plot(x1,y1,xlim=xrg)
> points(x2,y2,type='l',col='red')
```

all the x values are shown



Income vs Happiness

- Use the file happiness1000.RData
- Assume a linear model predicts happiness from income, and that a sample of 1000 participants in an experiment is contained in the file
- Plot the regression line that best fits the data
- Overlay this line with the scatter data
  (these two plots can be drawn in any order that suits you)
- What is the Pearson correlation coefficient between income and happiness
- What is the equation of the best line fit?
- What is the sum of residual squares?

```
# pause after each complete plot
#(after several plots if mfrow is used)

par(ask=T)
```

Pause execution

```
pause <- function(msg="") {
    cat("Return...[",msg,"]","\n")
    readline()
}
```

```
# read file, space-delimited
df = read.table("happiness1000.RData");
```

```
# check that data is read in properly
head(df)


# check the header labels
colnames(df)
```

```r
# compute linear model

m = lm(df$happiness ~ df$income)

# why not df$income ~ df$happiness?



# plot the data

plot(df$income, df$happiness)

abline(m, col='red')  # overlay regression line
```

```
# correlation coefficient

r = cor(df$income, df$happiness)

cat("correlation coefficient: ", r, "\n")
```

```
# what is slope and intercept?
coefs = m$coefficients
print(coefs)
pause("after coefs")
intercept = coefs[1]
slope = coefs[2]
cat("slope= ", slope, "\n")
cat("intercept= ", intercept, "\n")
pause("after slope/intercept")
```

Also provided by

```
   print(m)      # but must be transcribed manually
```

```
# sum of residual squares
# residual = (model fit) - (measured data)
# method 1
pause("residuals: method 1")
y = df$happiness
x = df$income
model = intercept + slope * x
residuals = (model-y)^2
sum.res = sum(residuals)
cat("sum of residuals: ", sum.res, "\n")
```

```
pause("residuals: method 2")
# method 2
sum.res = sum(m$residuals^2)
print(sum.res)
cat("sum(m$residuals^2)= ", sum.res, "\n")
```

Repeat with different data

- Repeat the previous experiment using the file happiness10.RData
- Use the regression line to estimate the happiness at an income level of 37, 48, 52, and 56.
- Repeat the above but impose that the line go through (zero happiness, zero income). Draw the scattergram and best line fit on the same plot. What is the correlation between income and happiness in this case? Is it lower or higher than the previous value? Why?

```
# pause after each complete plot (after several plots if mfrow is used)
par(ask=T)
#--------------------------
pause <- function(msg="") {
   cat("Return...[",msg,"]","\n")
   readline()
}
#--------------------------
# read file, space-delimited
df = read.table("happiness1000.RData");

# compute linear model
m = lm(df$happiness ~ df$income)
# why not df$income ~ df$happiness?

# plot the data
plot(df$income, df$happiness)
abline(m, col='red')  # overlay regression line

# correlation coefficient
r = cor(df$income, df$happiness)
cat("correlation coefficient: ", r, "\n")
```

```
# what is slope and intercept?
coefs = m$coefficients
print(coefs)
pause("after coefs")
intercept = coefs[1]
slope = coefs[2]
cat("slope= ", slope, "\n")
cat("intercept= ", intercept, "\n")
pause("after slope/intercept")

# sum of residual squares
# residual = (model fit) - (measured data)
# method 1
pause("residuals: method 1")
y = df$happiness
x = df$income
model = intercept + slope * x
residuals = (model-y)^2
sum.res = sum(residuals)
cat("sum of residuals: ", sum.res, "\n")

pause("residuals: method 2")
# method 2
sum.res = sum(m$residuals^2)
print(sum.res)
cat("sum(m$residuals^2)= ", sum.res, "\n")
```

Repeat previous code

Replace the file "happiness10.RData" by the file "happiness10000.RData"

Use the regression line to estimate the happiness at an income level of 37, 48, 52, and 56.

Model estimates

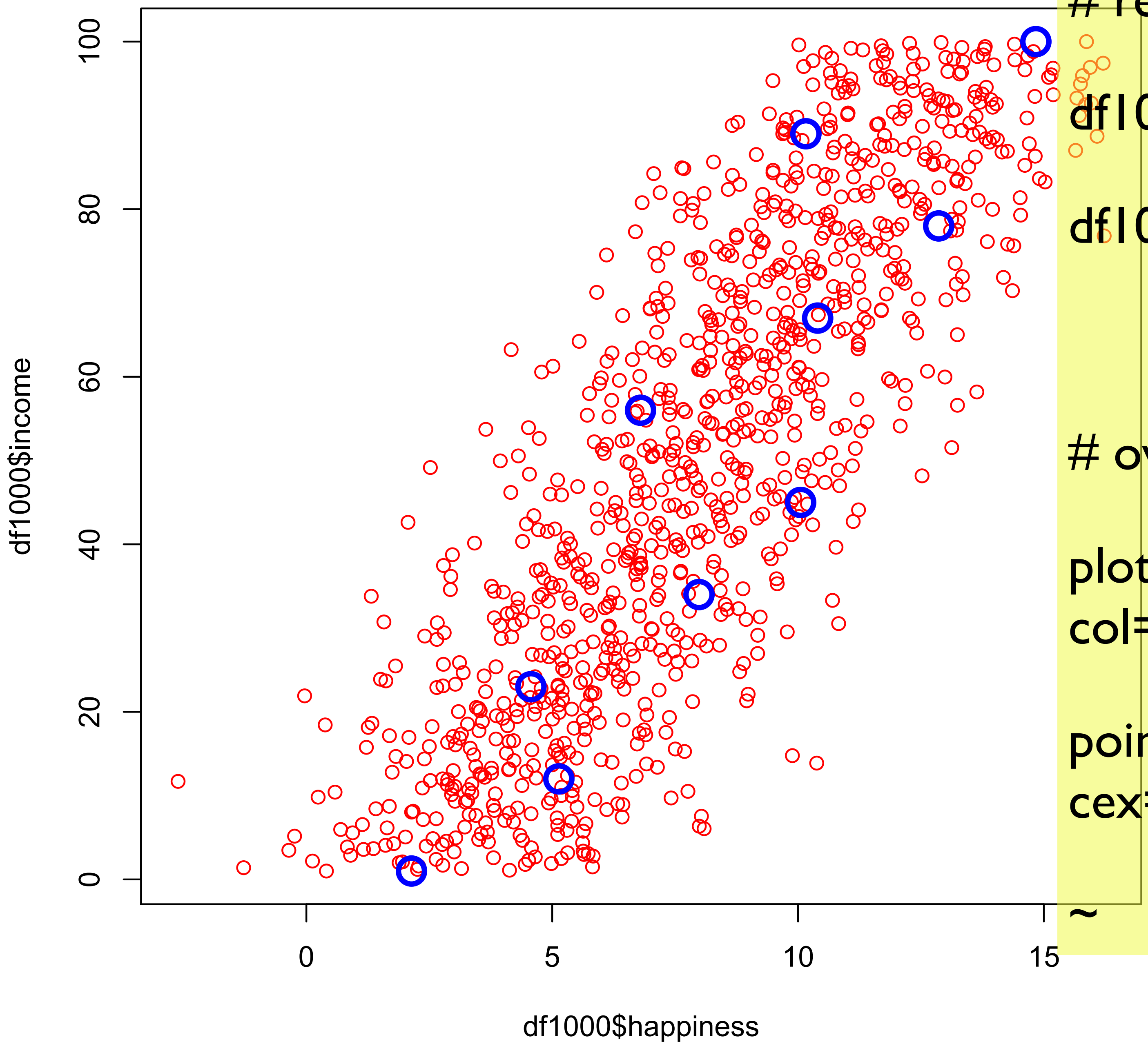
```
> values = c(37,48,52,56)

> predicted = intercept + values*slope
```

Correlation

```
# The correlation is based on the scattered data,
# not on the regression line.

# Therefore it is the same as when the regression
# does not contain "+0"
```

Both datasets

- Superimpose a scattergram of the data from happiness1000.RData in red, and a scattergram of the data from happiness10.RData in blue (make the blue symbols thicker and larger)



```
# read file, space-delimited

df10 = read.table("happiness10.RData");

df1000 = read.table("happiness1000.RData");



# overlay scattergrams

plot(df1000$happiness, df1000$income,
col='red')

points(df10$happiness, df10$income, col='blue',
cex=2, lw=3)

~
```

Create a function

- Create a function to solve the problem with one file (the function should take the file name as argument)
  - draw the scattergram + the regression line
  - print out the Pearson correlation coefficient
  - return the “linear model”
- Apply the function to each of the input files (with 10 and 1000 points)

regress_problem1_function.r

```
problem1 = function(file)
{
        # read file, space-delimited
        df = read.table(file)
        ....
}
```

Hints

- Use model = lm(x,y) to get the linear model
- Use summary(model) to get significance levels (p-value) for the slope and intercept

Maze_UniversityOfIllinois.csv

- I am interested in whether the women execute the maze differently than men (in terms of errors and timing)
  - Plot the scattergram of times for women in trial 1 against their time in trial 2. Do the same for men (superimpose on original plot in a different color). Create an additional plot between trials 1 and 15. What can one conclude? Create a final plot between trials 14 and 15.
- Plot the average time for women as a function of the average time for men (using all the trial data). What is the best linear fit to the data?
- Test whether Null Hypothesis (the mean time for ladies on a given trial is the same as the mean time for men (for each of the 15 trials). Use a function to ease the task.

Approach

- Similar to the previous problem

- Read data, keep only relevant columns (timings and gender)
- Perform analysis (t.test, shapiro.test, plot, hist, etc.)

read data

```
# comma-delimited
df = read.csv("Maze_UniversityOfIllinois.csv")


nbcols = ncol(df)
remove.cols = c(2,seq(7,nbcols,2))
df = df[,remove.cols]
~
```

```
# extract subsets
df.women = df[df$Gender == 'F',]
df.men = df[df$Gender == 'M',]

# scattergrams between 1st and 2nd trials
plot(T1T ~ T2T, data=df.women)
points(T1T ~ T2T, data=df.men, col='red')

#correlations
cor1.w = cor(df.women$T1T, df.women$T2T)
cor1.m = cor(df.men$T1T, df.men$T2T)
cat("correlation [1-2] (m/w): ", cor1.m, cor1.w, "\n")
```

cat revisited

Create the following script and run it in RStudio:

```
cat("my name is ")
cat("gordon")

cat("my name is\n")
cat("gordon\n")
```

```
my name is gordonmy name is
gordon
```

no carriage return

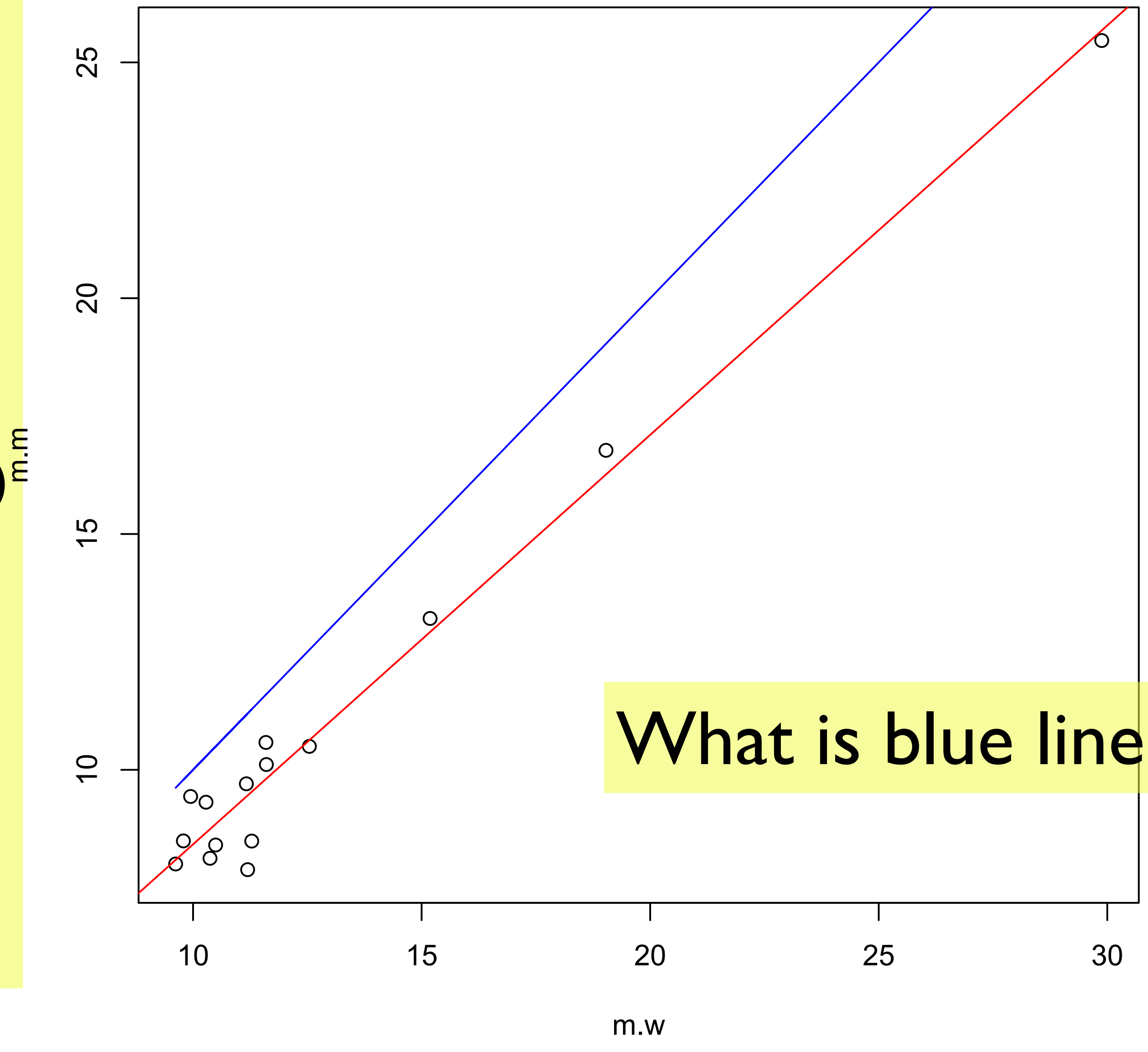
carriage return

Arguments to cat:

strings, numbers, vectors
(nothing else!!)

Plot the average time for women as a function of the average time for men (using all the trial data). What is the best linear fit to the data?

```
df.w = df.women[,-1]
df.m = df.men[,-1]

m.w = colMeans(df.w)
m.m = colMeans(df.m)

mo = lm(m.m ~ m.w)
plot(m.w, m.m)
abline(mo, col='red')
```

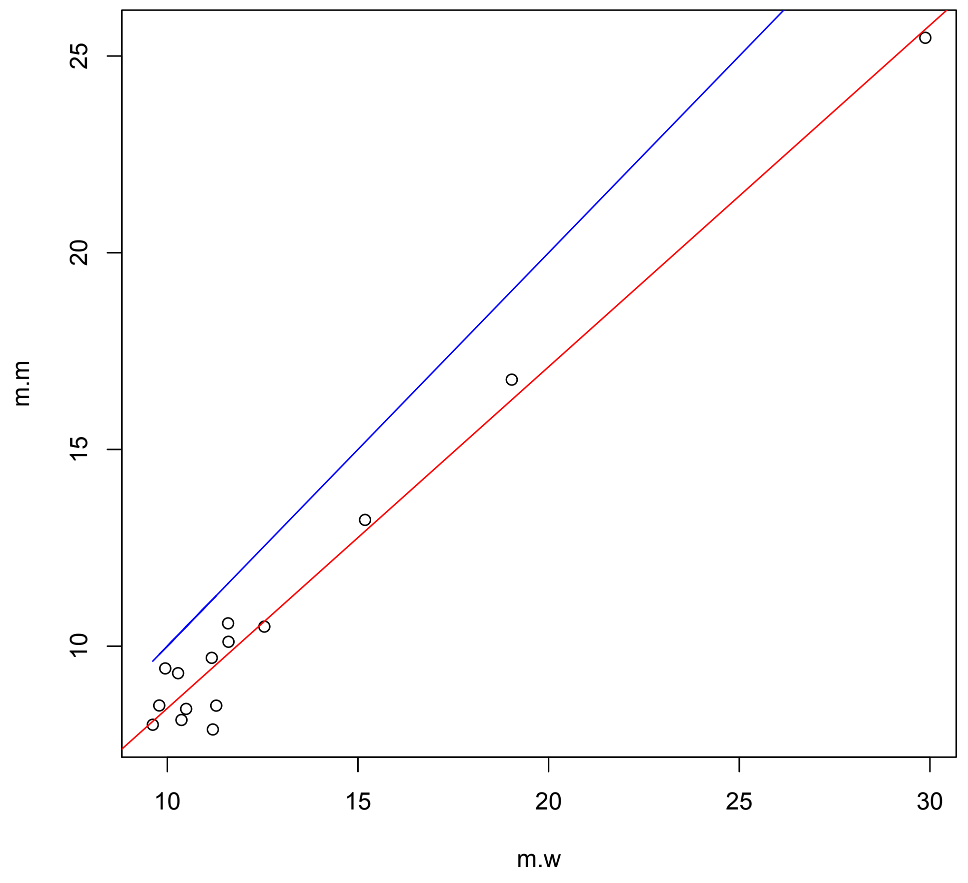


What is blue line?

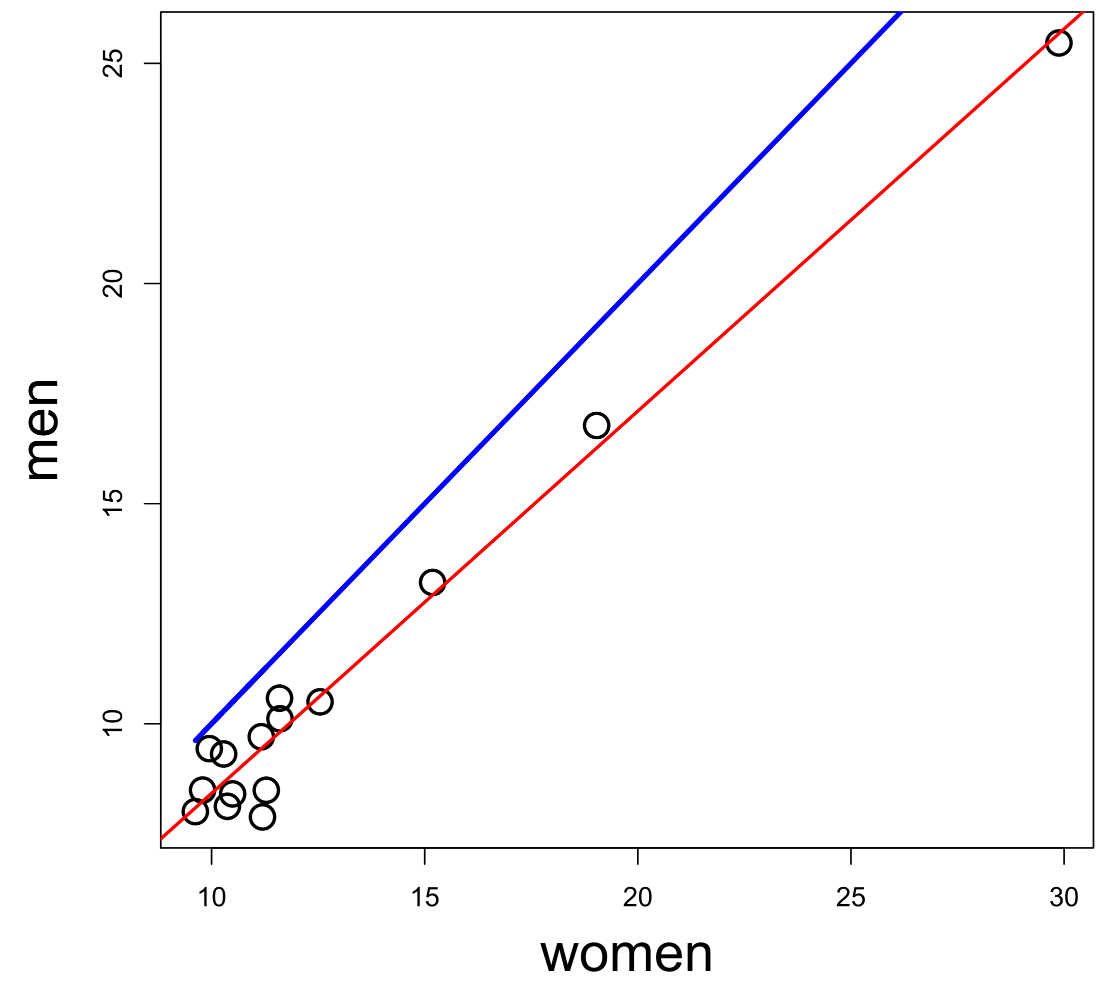
# One sees that the women run consistently
#slower than the men.

# superimpose a plot that assumes that
#women and men run at the same speed.

```
lines(m.w, m.w, col='blue')
```



Better plots



```
df.w = df.women[,-1]

df.m = df.men[,-1]

m.w = colMeans(df.w)

m.m = colMeans(df.m)

mo = lm(m.m ~ m.w)

plot(m.w, m.m, lwd=2, xlab='women',
ylab='men', cex=2, cex.lab=2)

abline(mo, col='red', lwd=2)

lines(m.w, m.w, col='blue', lwd = 3)
```

Test whether Null Hypothesis (the mean time for ladies on a given trial is the same as the mean time for men (for each of the 15 trials). Use a function to ease the task.

```
> # test for trial one

> t = t.test(df.m[,1], df.w[,1])

> print(t)
```

How to extract p-value for use inside an R-program?

```
        Welch Two Sample t-test


data:  df.m[, 1] and df.w[, 1]

t = -1.4267, df = 78.582, p-value = 0.1576

alternative hypothesis: true difference in means is not equal to 0

95 percent confidence interval:

 -10.576580   1.745424

sample estimates:

mean of x mean of y

 25.46335  29.87893
```

There is insufficient evidence to reject the null hypothesis.
The two means are within the 95 percent confidence interval of mean(df.m)-mean(df.w)

names(...)

```
t = t.test(df.m[,1], df.w[,1])
print(t)

> names(t)
[1] "statistic"   "parameter"   "p.value"     "conf.int"
"estimate"
[6] "null.value"  "alternative" "method"      "data.name"
```

```
> t$p.value

[1] 0.1576363
```

Process all trials

Create a function to process a single trial

```
null.hyp = function(trial)
{
    t = t.test(df.m[,trial], df.w[,trial])
    p.value = t$p.value
    cat("trial: ", trial, "   p: ", p.value, "\n")
}
```

Results for all 15 trials

```
null.hyp(1)
null.hyp(2)
null.hyp(3)
null.hyp(4)
null.hyp(5)
null.hyp(6)
null.hyp(7)
null.hyp(8)
null.hyp(9)
null.hyp(10)
null.hyp(11)
null.hyp(12)
null.hyp(13)
null.hyp(14)
null.hyp(15)
```

Lots of typing!

$\longrightarrow$

```
trial:  1   p:  0.1576363
trial:  2   p:  0.4604797
trial:  3   p:  0.1949634
trial:  4   p:  0.03571727
trial:  5   p:  0.2903947
trial:  6   p:  0.1052619
trial:  7   p:  0.06423717
trial:  8   p:  0.4662616
trial:  9   p:  0.255831
trial:  10   p:  0.01520369
trial:  11   p:  0.00841396
trial:  12   p:  0.01000343
trial:  13   p:  0.1220231
trial:  14   p:  0.02106974
trial:  15   p:  0.04624973
```

Better way: loops

```
for(i in 1:15)
{
   null.hyp(i)
}
```

$\longrightarrow$

```
trial:  1   p:  0.1576363
trial:  2   p:  0.4604797
trial:  3   p:  0.1949634
trial:  4   p:  0.03571727
trial:  5   p:  0.2903947
trial:  6   p:  0.1052619
trial:  7   p:  0.06423717
trial:  8   p:  0.4662616
trial:  9   p:  0.255831
trial:  10   p:  0.01520369
trial:  11   p:  0.00841396
trial:  12   p:  0.01000343
trial:  13   p:  0.1220231
trial:  14   p:  0.02106974
trial:  15   p:  0.04624973
```

More on Loops

http://faculty.washington.edu/kenrice/sisg/SISG-08-05.pdf

?Control

if(cond) expr

if(cond) cons.expr else alt.expr

for(var in seq) expr

while(cond) expr

repeat expr

break

next

?shapiro.test

Usage:

```
shapiro.test(x)
```

Arguments:

x: a numeric vector of data values. Missing values are allowed, but the number of non-missing values must be between 3 and 5000.

?fligner.test

Description:

Performs a Fligner-Killeen (median) test of the null that the variances in each of the groups (samples) are the same.

Usage:

```
fligner.test(x, ...)

## Default S3 method:
fligner.test(x, g, ...)

## S3 method for class 'formula'
fligner.test(formula, data, subset, na.action, ...)
```

Read the section on arguments

Extending data.frames

- Add the times of all 15 trials into a new column

```
total.time = df$T1T + df$T2T + ... + df$T15T

(lots of typing)  or

total.time = with(df, T1T+T2T+T3T ... + T15T)

    or
```

Sum of columns

```
> rowSums(df[,2:15])

 [1] 125.033 222.289 140.669 163.037 128.720 141.024 268.192 245.062 216.080 144.048 142.116
[12] 146.593 157.811 222.791 261.429 148.557 210.809 242.816 117.284 114.081 238.961 136.553
[23] 138.195 261.880 122.749 245.590 213.109 215.717 169.264 218.250 153.553 108.192 200.443
[34] 156.175 221.357  90.746 143.081 129.640 221.024 159.298 134.493 147.443 288.063 145.888
[45] 166.870 122.924 183.013 201.660 122.028  90.449 145.182 140.503 160.800 137.897 149.989
[56] 135.924 218.656 202.938 176.951  88.618 179.077 152.456 214.861 136.006  95.445 130.432
[67] 147.517 167.465 146.596 124.509 123.283 128.160 194.344 106.175 174.925 177.042 150.245
[78] 203.624 152.374 125.998 183.770 173.209 300.752 151.765 253.540 107.582 101.797 115.294
[89] 200.958 197.578 173.923 370.104
```

```
apply(df[,2:15],1,sum) # same result
```

```
> colSums(df[,2:15])

    T1T      T2T      T3T      T4T      T5T      T6T      T7T      T8T
T9T     T10T

2536.914 1642.632 1302.221 1055.827 1017.998  995.965
957.084  890.503  899.479  865.211

   T11T     T12T     T13T     T14T

 903.947  846.283  838.332  870.917
```

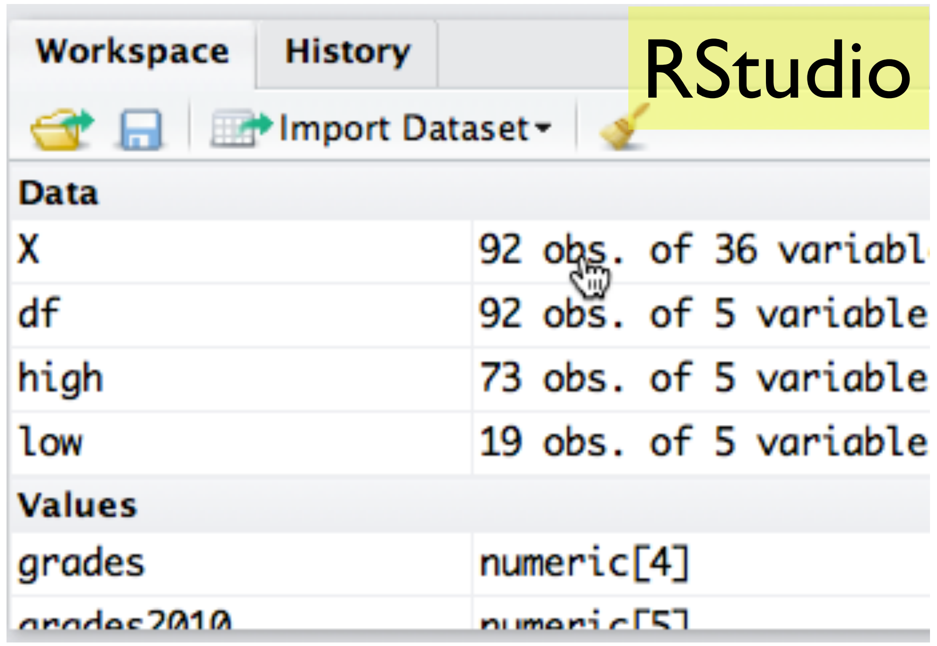
Every column is summed up
(NOT WHAT WE WANT!!)

Add new column to data.frame

```
col = apply(sum,1,df[,2:15])
df = data.frame(df, total.time=col)
```

List variables in workspace

```
> ls()

 [1] "df"        "grades"      "grades2010"
"grades2011" "grades2012" "high"

 [7] "low"       "m"          "x"          "X"
"xx"         "y"

[13] "y1"        "y2"
```



Data

X	92 obs. of 36 variabl
df	92 obs. of 5 variable
high	73 obs. of 5 variable
low	19 obs. of 5 variable

Values

grades	numeric[4]
grades2010	numeric[5]