Introduction to Statistics with R

Session R02: Correlation

Marvin Schmitt

The example data set

We analyze structures in the **Chicago Face Database** (Ma et al., 2015). Each row refers to a portrait which was rated with respect to different categories by a sample of raters.

```
df = read.csv("R02_notes_dataset.csv")
nrow(df)
```

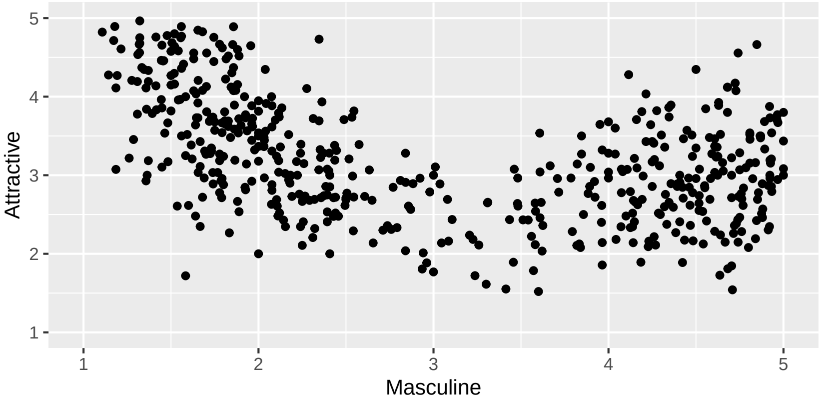
```
## [1] 597
```

```
colnames(df)
```

```
##  [1] "ID"            "Gender"        "Age"           "Afraid"
##  [5] "Angry"         "Attractive"    "Babyface"      "Disgusted"
##  [9] "Dominant"      "Feminine"      "Happy"         "Masculine"
## [13] "Prototypic"    "Sad"           "Surprised"     "Threatening"
## [17] "Trustworthy"   "Unusual"       "Nose_Width"    "Nose_Length"
## [21] "FaceRoundness" "Noseshape"
```

Scatterplots

```
ggplot(df, aes(x=Masculine, y=Attractive))+
geom_point() +
ylim(1,5) + xlim(1,5)
```



Digression I: `filter`

```
df %>% filter(Gender=="M")
```

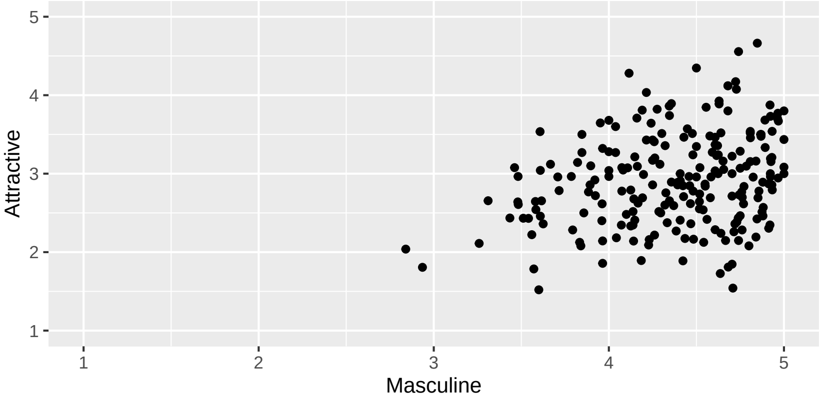
```
##      ID Gender      Age   Afraid    Angry Attractive Babyface Disgusted
## 1    58      M 23.80000 1.240000 2.520000   3.040000 2.625000  2.040000
## 2    59      M 26.22222 1.846154 2.888889   2.777778 2.296296  2.296296
## 3    60      M 26.54167 1.708333 1.583333   2.458333 3.041667  1.478261
## 4    61      M 23.91667 1.833333 1.625000   3.041667 3.625000  1.347826
## 5    62      M 34.83333 1.916667 2.130435   2.625000 2.083333  1.708333
## 6    63      M 26.92857 1.785714 2.678571   2.285714 2.428571  2.107143
## 7    64      M 25.91667 1.652174 1.708333   2.958333 2.416667  1.500000
## 8    65      M 23.32143 1.428571 1.535714   2.964286 2.392857  1.750000
## 9    66      M 27.75000 1.592593 1.428571   2.857143 2.678571  1.214286
## 10   67      M 21.04762 2.000000 2.250000   3.809524 3.700000  1.952381
## 11   68      M 26.32143 1.333333 1.851852   3.428571 2.148148  1.321429
## 12   69      M 25.55556 1.925926 3.222222   1.846154 3.307692  2.259259
## 13   70      M 28.07143 1.785714 2.321429   2.142857 2.285714  2.111111
## 14   71      M 29.22222 1.925926 2.259259   3.888889 2.592593  1.814815
## 15   72      M 56.38462 1.884615 1.518519   3.076923 1.851852  2.037037
## 16   73      M 28.48000 1.240000 1.720000   4.120000 2.875000  1.520000
## 17   74      M 43.00000 1.615385 3.076923   2.538462 1.307692  2.538462
## 18   75      M 33.59259 1.250000 1.357143   3.214286 1.964286  1.535714
## 19   76      M 36.90000 1.933333 3.900000   2.517241 1.533333  3.033333
## 20   77      M 23.79310 1.793103 2.862069   2.344828 2.758621  2.357143
## 21   78      M 24.11538 1.884615 2.423077   3.230769 2.846154  2.461538
## 22   79      M 21.92593 1.740741 1.962963   2.222222 2.000000  1.629630
## 23   80      M 28.96000 1.480000 3.040000   1.520000 2.040000  2.800000
## 24   81      M 24.68000 1.920000 2.080000   3.520000 2.840000  2.040000
## 25   82      M 46.81818 1.909091 2.590909   1.809524 2.090909  1.818182
## 26   83      M 41.42308 1.400000 2.076923   2.384615 1.538462  1.384615
## 27   84      M 25.32143 1.642857 2.222222   3.428571 2.642857  1.821429
```

Digression II: select

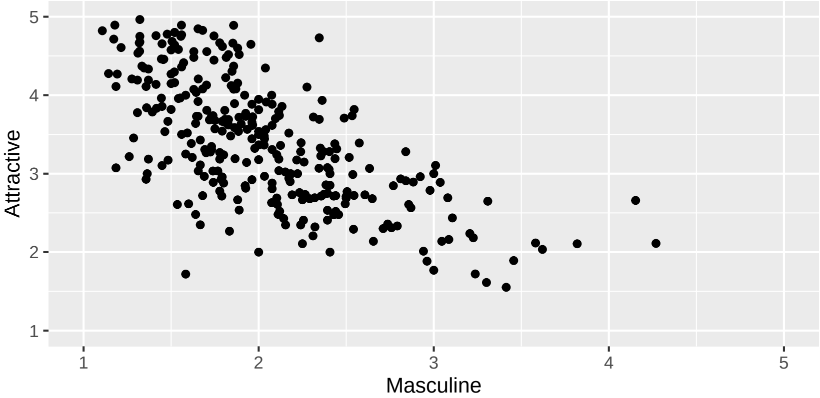
```
df %>%
  select(Attractive)
```

```
##     Attractive
## 1     4.111111
## 2     3.111111
## 3     3.000000
## 4     3.275862
## 5     3.172414
## 6     4.333333
## 7     2.714286
## 8     2.137931
## 9     3.038462
## 10    4.080000
## 11    2.615385
## 12    3.307692
## 13    2.607143
## 14    3.000000
## 15    3.730769
## 16    2.814815
## 17    3.266667
## 18    3.833333
## 19    4.678571
## 20    2.928571
## 21    3.173913
## 22    3.000000
## 23    3.689655
## 24    3.185185
## 25    3.240000
## 26    3.034483
```

```
df %>% filter(Gender=="M") %>%
  ggplot(., aes(x=Masculine, y=Attractive))+
  geom_point() +
  ylim(1,5) + xlim(1,5)
```

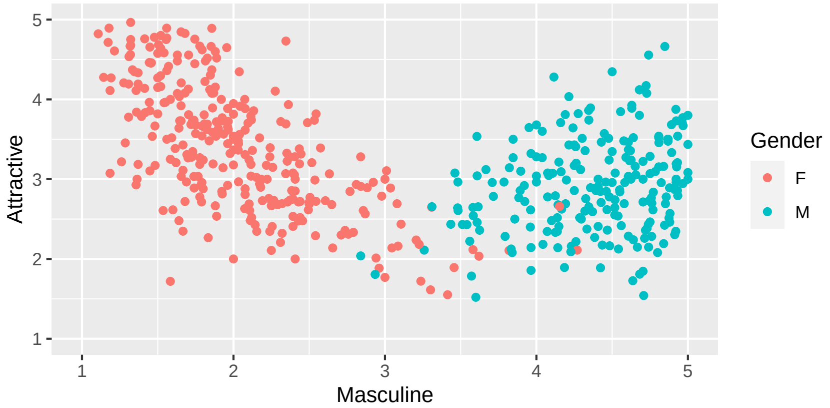


```r
df %>% filter(Gender=="F") %>%
  ggplot(., aes(x=Masculine, y=Attractive))+
  geom_point() +
  ylim(1,5) + xlim(1,5)
```



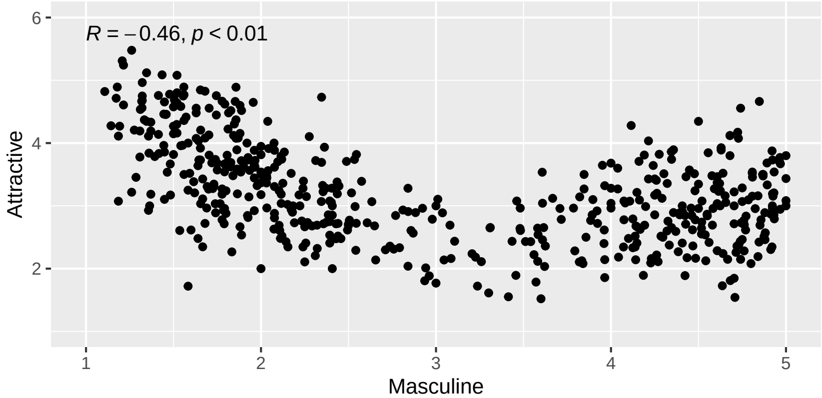
Scatterplots: Group variables

```
ggplot(df, aes(x = Masculine, y = Attractive, color=Gender))+
  geom_point() +
  ylim(1, 5) + xlim(1, 5)
```

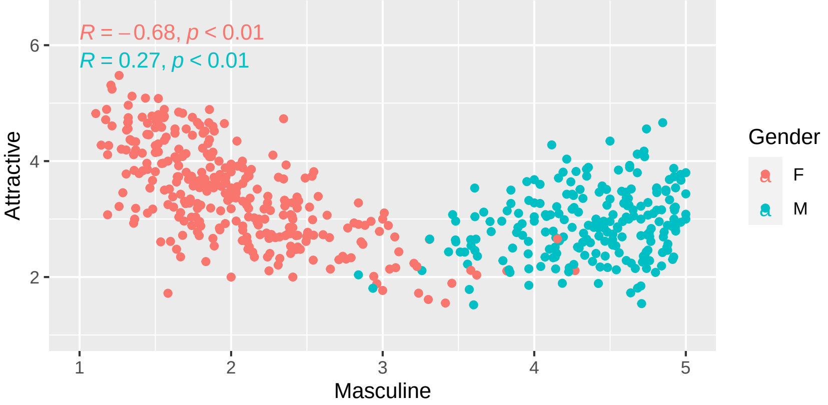


Scatterplots: Print correlation coefficients

```
ggplot(df, aes(x=Masculine, y=Attractive))+
geom_point() +
ylim(1, 6) + xlim(1, 5) +
stat_cor(p.accuracy = 0.01)
```



```
ggplot(df, aes(x=Masculine, y=Attractive, color=Gender))+
geom_point() +
ylim(1, 6.5) + xlim(1, 5) +
stat_cor(p.accuracy = 0.01)
```



Correlation matrices: Computation

```r
df %>%
  select(Angry, Disgusted, Happy, Sad, Surprised, Attractive, Threatening) %>%
  cor() %>%
  round(3)
```

```
##              Angry Disgusted  Happy    Sad Surprised Attractive Threatening
## Angry        1.000     0.843 -0.606  0.454     0.082     -0.302       0.834
## Disgusted    0.843     1.000 -0.536  0.572     0.206     -0.296       0.687
## Happy       -0.606    -0.536  1.000 -0.573     0.189      0.463      -0.449
## Sad          0.454     0.572 -0.573  1.000     0.131     -0.323       0.272
## Surprised    0.082     0.206  0.189  0.131     1.000      0.050       0.117
## Attractive  -0.302    -0.296  0.463 -0.323     0.050      1.000      -0.375
## Threatening  0.834     0.687 -0.449  0.272     0.117     -0.375       1.000
```

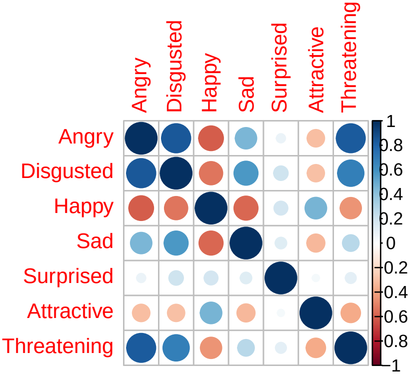
Correlation matrices: Visualization

We can visually encode the entries in a **correlation matrix** while maintaining the matrix structure:

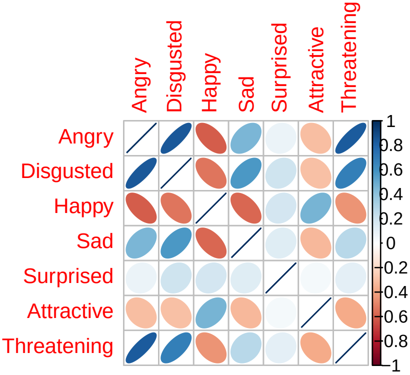
- Color coding
- Circles
- Ellipses
- Correlation coefficients r_{xy}
- ...

We can use the function `corrplot::corrplot` to visualize correlation matrices.

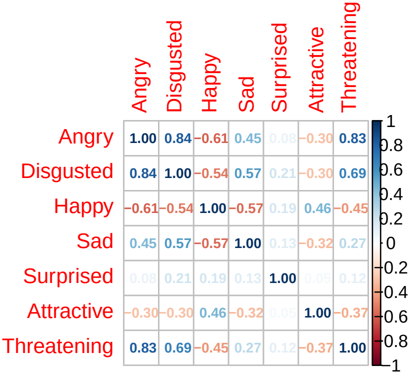
```
df %>%
  select(Angry, Disgusted, Happy, Sad, Surprised, Attractive, Threatening) %>%
  cor() %>%
  corrplot::corrplot()
```



```
df %>%
  select(Angry, Disgusted, Happy, Sad, Surprised, Attractive, Threatening) %>%
  cor() %>%
  corrplot::corrplot(method="ellipse")
```



```r
df %>%
  select(Angry, Disgusted, Happy, Sad, Surprised, Attractive, Threatening) %>%
  cor() %>%
  corrplot::corrplot(method="number", number.cex = 0.7)
```



Important arguments for `corrplot::corrplot`:

- `method`: cell content
- `tl.col='black'`: black labels
- `cl.pos='n'`: remove colorbar
- `type`: display entire/lower/upper matrix
- `number.cex`: size of numbers if `method=number`

Testing assumptions

In order to ease interpretation of a correlation coefficient r_{xy}, the variables X, Y should be:

- **metric**,
- **normal**, and
- **homoscedastic**.

We can use the function `shapiro.test(x)` to test `x` for **normality**.

Interpretation:

- if $p \geq .05$: normality assumption not violated
- if $p < .05$: normality assumption violated

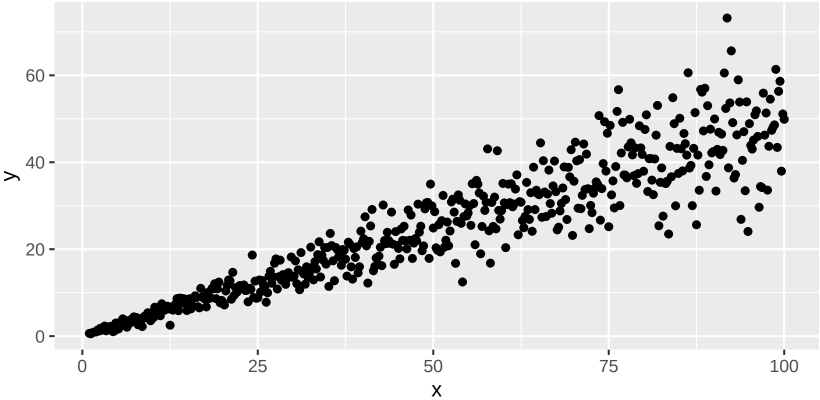
```
shapiro.test(df$Trustworthy)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df$Trustworthy
## W = 0.99712, p-value = 0.3776
```

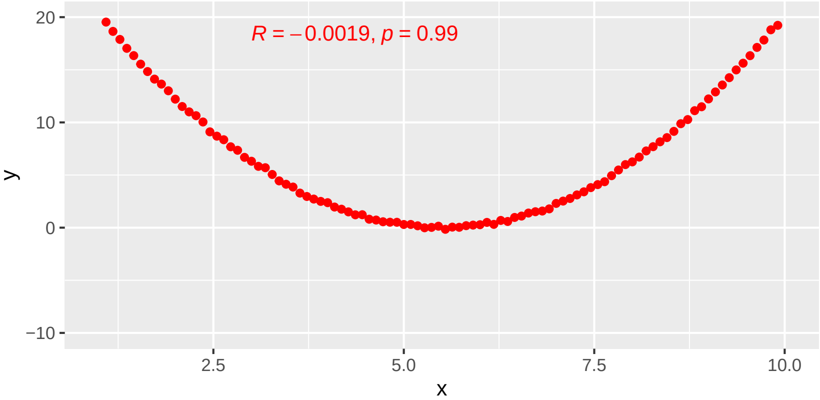
```
shapiro.test(df$Attractive)
```

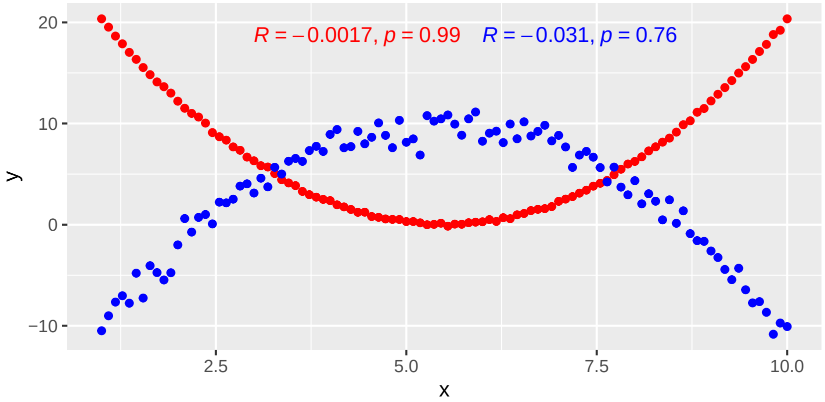
```
##
##  Shapiro-Wilk normality test
##
## data:  df$Attractive
## W = 0.98357, p-value = 0.000002952
```

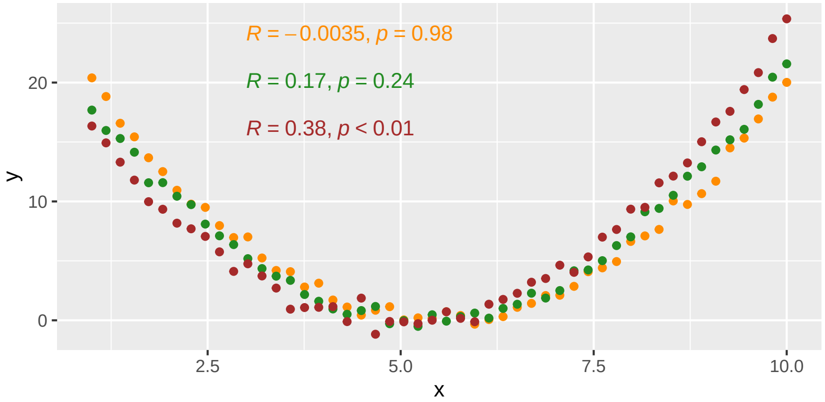
The scatterplot allows for a visual inspection of **homoscedasticity**:

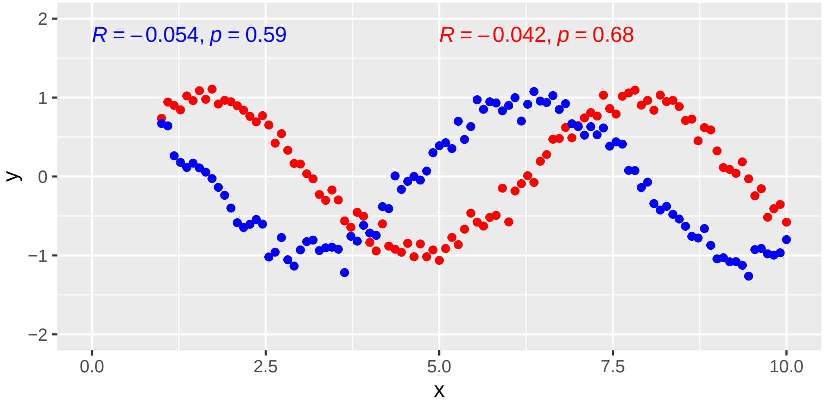


Another caveat:




R = −0.0017, p = 0.99
R = −0.031, p = 0.76
20
10
0
−10
y
2.5
5.0
7.5
10.0
x


$R = -0.0035$, $p = 0.98$
$R = 0.17$, $p = 0.24$
$R = 0.38$, $p < 0.01$
y
20
10
0
2.5
5.0
7.5
10.0
x



R = −0.054, p = 0.59
R = −0.042, p = 0.68
2
1
0
−1
−2
y
0.0
2.5
5.0
7.5
10.0
x