# R Statistical Software in Action for Newcomers
# Data Analysis

Arndt Leininger

@a_leininger
arndt.leininger@fu-berlin.de

24 April 2018

# Program

**Tuesday, 24 April 2018**

09:00h - 10:30h Data Analysis

*10:30h - 10:45h break*

10:45h - 12:15h Data visualization

12:15h - 12:30h Concluding remarks

# Correlation and difference-in-means

# Correlation

```r
library(foreign)
d <- read.dta("../data/EUsuppDK.dta")

cor(d$age, d$left_right)
```

```
## [1] NA
```

```r
cor(d$age, d$left_right, use = "complete.obs")
```

```
## [1] 0.07708635
```

# T-Test

```r
t.test(left_right ~ sex, d)
```

```
##
##  Welch Two Sample t-test
##
## data:  left_right by sex
## t = 2.3134, df = 955.45, p-value = 0.02091
## alternative hypothesis: true difference in means is not equal
## 95 percent confidence interval:
##  0.04020583 0.48985106
## sample estimates:
##   mean in group male mean in group female
##             5.568826             5.303797
```

- Missings (`NA`) are dropped automatically

# T-Test

```r
t.test(left_right ~ sex, d, var.equal = T)
```

```
##
##  Two Sample t-test
##
## data:  left_right by sex
## t = 2.3064, df = 966, p-value = 0.0213
## alternative hypothesis: true difference in means is not equal
## 95 percent confidence interval:
##  0.03952888 0.49052801
## sample estimates:
##   mean in group male mean in group female
##             5.568826             5.303797
```

# Simple regression models

# Regression

- Regression models are functions that are fed an equation and data
- Further options are possible but optional
- The dependent variable is seperated by a tilde from the independent variables
- Equation: `dv ~ iv`
- No need for $ operator in the equation

```r
lm(left_right ~ age, d)
```

```
##
## Call:
## lm(formula = left_right ~ age, data = d)
##
## Coefficients:
## (Intercept)          age
##    5.087151     0.007995
```

# Formulae

**A complete overview of formulae in R**

https://ww2.coastal.edu/kingw/statistics/R-tutorials/formulae.html

# Model objects

- You can save the output of the `lm()` function just like with any other function.

```r
m1 <- lm(left_right ~ age, d)
```

- `lm()` is for linear models, i.e. OLS

# Model objects

- Once you save an estimated model as object you can always access it to obtain model statistics.

```
summary(m1)  # estimation results

coef(m1)  # coefficients

vcov(m1)  # Variance-Covariance Matrix

predict(m1)  # Predicted values

resid(m1)  # Residuals
```

# Model objects

```
summary(m1)  # estimation results
```

```
##
## Call:
## lm(formula = left_right ~ age, data = d)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.687 -1.303 -0.351  1.515  4.649
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.087151   0.157297  32.341   <2e-16 ***
## age         0.007995   0.003327   2.403   0.0164 *
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.787 on 966 degrees of freedom
##   (33 observations deleted due to missingness)
```

# A little trick

```r
summary(m <- lm(left_right ~ age, d))
```

```
##
## Call:
## lm(formula = left_right ~ age, data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.687 -1.303 -0.351  1.515  4.649
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.087151   0.157297  32.341   <2e-16 ***
## age         0.007995   0.003327   2.403   0.0164 *
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.787 on 966 degrees of freedom
##   (33 observations deleted due to missingness)
```

# A little trick

```
# but
summary(m = lm(left_right ~ age, d))
```

```
## Error in summary.lm(m = lm(left_right ~ age, d)): argument "o
```

# Model objects

```r
coef(m1)  # coefficients
```

```
## (Intercept)         age
## 5.087151366 0.007995059
```

# Model objects

```r
vcov(m1)  # Variance-Covariance Matrix
```

```
##               (Intercept)           age
## (Intercept)  0.0247424372 -4.872123e-04
## age         -0.0004872123  1.106937e-05
```

# Model objects

```r
predict(m1)  # Predicted values
```

```
##        1        2        3        4        5        6 
## 5.414949 5.422944 5.215072 5.303018 5.279033 5.223067 
##        7        8        9       10       11       12 
## 5.231062 5.295023 5.295023 5.303018 5.231062 5.414949 
##       13       14       15       16       17       18 
## 5.311013 5.303018 5.279033 5.239057 5.319008 5.279033 
##       19       20       21       22       23       24 
## 5.430939 5.422944 5.327003 5.247053 5.438934 5.446929 
##       25       26       27       28       29       30 
## 5.430939 5.271038 5.255048 5.263043 5.454924 5.462919 
##       31       32       33       34       35       36 
## 5.470914 5.311013 5.319008 5.327003 5.438934 5.334998 
##       37       38       39       40       41       42 
## 5.334998 5.271038 5.223067 5.478909 5.446929 5.342993 
##       43       44       45       46       47       48 
## 5.454924 5.350988 5.358983 5.486904 5.462919 5.494899 
##       49       50       51       52       53       54 
## 5.279033 5.366978 5.342993 5.470914 5.502894 5.350988 
```

# Model objects

```r
resid(m1)  # Residuals
```

```
##          1          2          3          4          5
## -0.4149488 -0.4229438 -0.2150723  1.6969820  2.7209672
##          6          7          8          9         10
## -1.2230674 -0.2310624 -0.2950229 -0.2950229 -0.3030180
##         11         12         13         14         15
## -0.2310624  1.5850512  0.6889870  1.6969820 -3.2790328
##         16         17         18         19         20
##  1.7609425  2.6809919  2.7209672 -1.4309389 -0.4229438
##         21         22         23         24         25
## -1.3270031 -0.2470525  1.5610660 -0.4469290  1.5690611
##         26         27         28         29         30
## -3.2710377 -2.2550476 -3.2630427 -0.4549241  1.5370809
##         31         32         33         34         35
## -0.4709142 -0.3110130  1.6809919 -3.3270031  2.5610660
##         36         37         38         39         40
##  0.6650018 -0.3349982 -3.2710377 -0.2230674  2.5210907
##         41         42         43         44         45
##  1.5530710 -1.3429932  0.5450759 -0.3509883  2.6410166
```

# Model objects

```r
class(m1)
```

```
## [1] "lm"
```

```r
objects(m1)
```

```
##  [1] "assign"        "call"          "coefficients" 
##  [4] "df.residual"   "effects"       "fitted.values"
##  [7] "model"         "na.action"     "qr"           
## [10] "rank"          "residuals"     "terms"        
## [13] "xlevels"
```

# Interactions

- Interactions can be specified as follows
  - `var1*var2` = `var1 + var2 + var1:var2`
  - `var1:var2` is simply the interaction term

```
m_i <- lm(left_right ~ sex * age, d)
# or
m_i <- lm(left_right ~ sex + age + sex:age, d)
```

# Polynomials

```
lm(left_right ~ age + I(age^2), d)  # second-order polynomial

lm(left_right ~ age + age^2, d)  # does not work
```

- `?I()`: “Change the class of an object to indicate that it should be treated ‘as is’.”

# A little trick

- Put the assignment of a model to an object in a `summary()` call to assign and view results at the same time

```
summary(m2 <- lm(left_right ~ age + I(age^2), d))
```

```
## 
## Call:
## lm(formula = left_right ~ age + I(age^2), data = d)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -4.7388 -1.3154 -0.3393  1.5501  4.6607 
## 
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  5.3107878  0.3941342  13.475   <2e-16 ***
## age         -0.0029424  0.0179840  -0.164    0.870    
## I(age^2)     0.0001153  0.0001863   0.619    0.536    
## ---
## Signif. codes:
```

# Time-Series analysis

- Lags and leads can be easily included in time-series data
- For this make sure that the data are sorted on the time variable
  - Remember to use `order()` not ‘sort()“(which is for vectors) to sort a data.frame
  - Or use `dplyr`’s `arrange()`
- `lag()` and `lead()`
- For second lag: `lag(var, 2)`

```
summary(m_l <- lm(y ~ x + lag(x) + lag(x, 2), d))
```

# Logistic regression

- Generalized linear models, such as probit or logistic regression, are provided through the `glm()` function

```
d$wealthy <- ifelse(d$wealth == "++" | d$wealth == "+",
    T, F)

# Probit
m_g <- glm(wealthy ~ age + sex, data = d)

# Logistic
m_g <- glm(wealthy ~ age + sex, family = binomial(), data = d)
```

# Predicted values

- `predict()` is a generic function to create predictions from various models

```
predicted_values <- predict(m1)
# but...
d$yhat <- predict(m1)
```

```
## Error in `$<-.data.frame`(`*tmp*`, yhat, value = structure(c(
```

```
# Error in `$<-.data.frame`(`*tmp*`, 'yhat', value =
# c(5.41494877919278, : replacement has 968 rows, data
# has 1001
```

# Predicted values

```r
d$yhat <- predict(m1, newdata = d)
# no error message because now predictions are also made
# for deleted observations; these predictions are
# obviously NA
```

# Predicted values

Or. . .

```
install.packages("broom")
```

```
library(broom)
augment(m1) %>% head
```

```
##   .rownames left_right age  .fitted    .se.fit
## 1         1          5  41 5.414949 0.05829785
## 2         2          5  42 5.422944 0.05781844
## 3         3          5  16 5.215072 0.10947787
## 4         4          7  27 5.303018 0.08063835
## 5         5          8  24 5.279033 0.08793296
## 6         6          4  17 5.223067 0.10665959
##       .resid        .hat   .sigma      .cooksd
## 1 -0.4149488 0.001064565 1.787636 2.876892e-05
## 2 -0.4229438 0.001047128 1.787634 2.939764e-05
## 3 -0.2150723 0.003754221 1.787673 2.740268e-05
## 4  1.6969820 0.002036810 1.786850 9.223870e-04
## 5  2.7209672 0.002421980 1.785534 2.822023e-03
```

# Predicted values

```r
df <- data.frame(age = min(d$age):max(d$age))

predict(m1, newdata = df)
```

```
##        1        2        3        4        5        6
## 5.207077 5.215072 5.223067 5.231062 5.239057 5.247053
##        7        8        9       10       11       12
## 5.255048 5.263043 5.271038 5.279033 5.287028 5.295023
##       13       14       15       16       17       18
## 5.303018 5.311013 5.319008 5.327003 5.334998 5.342993
##       19       20       21       22       23       24
## 5.350988 5.358983 5.366978 5.374973 5.382969 5.390964
##       25       26       27       28       29       30
## 5.398959 5.406954 5.414949 5.422944 5.430939 5.438934
##       31       32       33       34       35       36
## 5.446929 5.454924 5.462919 5.470914 5.478909 5.486904
##       37       38       39       40       41       42
## 5.494899 5.502894 5.510889 5.518885 5.526880 5.534875
##       43       44       45       46       47       48
## 5.542870 5.550865 5.558860 5.566855 5.574850 5.582845
```

# Hands-on I

# Hands-on I

https://gitlab.com/arndtl/r_workshop

# Presenting Results with R

# Lists

- Lists can contain any kind of objects of any type.
- Note: data.frames can also contain vectors of any of the three types but the vectors are forced to be of the same length.
- Example: One could have a list of differently sized vectors

```r
v1 <- c(1, 2, 3)
v2 <- c("a", "b", "c", "d")
alist <- list(v1, v2)
alist
```

```
## [[1]]
## [1] 1 2 3
##
## [[2]]
## [1] "a" "b" "c" "d"
```

# Tables

Using packages such as stargazer or texreg we can create nice regression tables.

```r
library(stargazer)
library(texreg)
```

# Tables

Here's an example in tex using stargazer.

```
m1 <- lm(mpg ~ cyl, mtcars)
m2 <- lm(mpg ~ cyl + gear, mtcars)

stargazer(list(m1, m2), header = F, float = F,
          font.size = 'tiny', single.row = T)
```

| | *Dependent variable:* | |
|---|---|---|
| | mpg | |
| | (1) | (2) |
| cyl | $-2.876^{***}$ (0.322) | $-2.743^{***}$ (0.373) |
| gear | | 0.652 (0.904) |
| Constant | $37.885^{***}$ (2.074) | $34.659^{***}$ (4.937) |
| Observations | 32 | 32 |
| $R^2$ | 0.726 | 0.731 |
| Adjusted $R^2$ | 0.717 | 0.712 |
| Residual Std. Error | 3.206 (df = 30) | 3.232 (df = 29) |
| F Statistic | $79.561^{***}$ (df = 1; 30) | $39.404^{***}$ (df = 2; 29) |
| *Note:* | | $^{*}$p<0.1; $^{**}$p<0.05; $^{***}$p<0.01 |

# Tables

stargazer provides tables in text, html and tex.

```
# output as text file
stargazer(m1, type = "text", out = "tables/m1.txt")
# output as html file which Word can read
stargazer(m1, type = "html", out = "tables/m1.html")
# output as tex, the default
stargazer(m1, out = "tables/m1.tex")
```

# Screenreg

- texreg's screenreg() function is very useful to quickly view some models.

```
screenreg(list(m1, m2))
```

```
##
## =================================
##              Model 1    Model 2
## ---------------------------------
## (Intercept)  37.88 ***  34.66 ***
##              (2.07)     (4.94)
## cyl          -2.88 ***  -2.74 ***
##              (0.32)     (0.37)
## gear                     0.65
##                         (0.90)
## ---------------------------------
## R^2           0.73       0.73
## Adj. R^2      0.72       0.71
## Num. obs.    32         32
## RMSE          3.21       3.23
```

# Tables

`texreg` provides tables `html`, `tex` and to screen.

```r
# output as tex file
texreg(m1, file = "tables/m1.tex")
# output as html file which Word can read
htmlreg(m1, file = "tables/m1.html")
# output as tex, the default
screenreg(m1)
```

# Regression tables for Word

### Via HTML

- Export to HTML using `htmlreg()` (package `texreg`) or `stargazer(..., type = 'html')`
- Then copy and paste to Word
- Or, better, link to the html file from within the Word Document
  - Word: Insert -> Object (dropwdown) -> Text from File -> Insert (dropdown) -> Insert as link; hit F9 to refresh
  - LibreOffice Write: Insert -> Section -> Check option “Link” and choose document; to refresh: Edit -> Links -> click “Update”
  - http://www.techrepublic.com/article/link-to-another-file-in-your-word-document/

# Important packages for regression analysis

- `lmtest` for F-test and other tests
- `tseries` for time-series analysis
- `plm` for panel data analysis
- `lme4` for multilevel models
- `MatchIt` and `Matching` for matching

# Hands-on II

# Hands-on II

https://gitlab.com/arndtl/r_workshop

# Further packages

# ReporterRs

- ReporterRs is a package allows creation of entire(!) Word and Power Point documents
- this includes (regression) tables

```
install.packages("ReporteRs")
```

http://davidgohel.github.io/ReporteRs/index.html

# Regression tables with ReporterRs

```r
library(ReporteRs)

# save the model in a data.frame
mdata <- as.data.frame(summary(m1)$coefficients)

# Define significance cutoffs
signif.codes <- cut(mdata[,4],
                    breaks = c( -Inf, 0.001, 0.01, 0.05, Inf),
                    labels= c("***", "**", "*", "" ) )

#format the values of coefficients, etc.
mdata[, 1:3] <- apply(mdata[, 1:3], 2, round, 2)
mdata[, 4] <- ifelse(mdata[, 4] < .001, "< 0.001",
                     round(mdata[, 4], 3))
# add signif codes to data
mdata$Signif = signif.codes
```

# Regression tables with ReporterRs

```r
mdata

# create an empty FlexTable
coef_ft = FlexTable(data = mdata, add.rownames=TRUE,
                    body.par.props = parRight(),
                    header.text.props = textBold(),
                    header.columns = T)
```

# Regression tables with ReporterRs

```r
# format the table a bit
coef_ft = setFlexTableBorders(coef_ft,
                               inner.vertical = borderNone(),
                               inner.horizontal = borderNone(),
                               outer.vertical = borderNone(),
                               outer.horizontal = borderSolid())
```

# Regression tables with ReporteRs

Now we'll save the table in a Word document.

```r
# Create an empty document
doc <- docx()

# Add the regression table to the document
doc <- addFlexTable(doc, coef_ft)

# Save the document to the hard drive
writeDoc(doc, file = "../tables/regtable.docx")
```