Help

```
?
??
example()
apropos()
help.search()
```

Packages

In order to use functionalities from a certain package, we need to first install and then load the package:

```
# install package (do once):
install.packages("")

# load package (do once in every script):
require(); library()
```

Formula syntax

Most of the functions that we need for this course uses a formula syntax:

```
goal(y ~ x | z, data = mydata, ...)
```

where `goal` may be a function for plotting, calculating numerical summaries or making inference.

For plots:

- `y` is y-axis variable
- `x` is x-axis variable
- `z` a conditioning variable (separate panels).

For other things:

'`y ~ x | z`' can usually be read '`y` is modeled by (or depends on) `x` differently for each `z`'.

Numerical summaries

These functions from the mosaic-package uses a formula syntax.

```
favstats()      # min/max, median, etc.
tally()         # tabulate data
mean()
median()
sd()            # standard deviation
cor()           # correlation
```

General graphics

```
gf_boxplot()
gf_point()      # scatter plot
gf_histogram()
gf_bar()        # bar graph

mplot(HELPrct)# different plots
splom()         # matrix of scatter plots
```

Distributions

```
plotDist()      # plot theoretical distribution
pdist()         # find prob. from percentile
qdist()         # find percentile from prob.
```

Hypothesis tests

```
t.test()        # t-test
binom.test()    # binomial (exact) test
prop.test()     # approximate test
fisher.test()   # Fisher's exact test
cor.test()      # correlation test
chisq.test()    # chi-square test
```

Linear regression

```
model <- lm()     # fit linear model
summary(model)  # model fit summary
coef(model)     # estimated parameters
confint(model)  # CI for estimates
anova(model)    # F-tests, etc.
drop1(model)
rstudent(model) # Studentized residuals
fitted(model)   # fitted values
plotModel(model)# plot regression lines
```

```
model <- glm()    # generalized linear model
```

Data

```
# Load data:
read.file(); read.delim(); read.csv()
```

```
# Data information
nrow(); ncol()  # data dimensions
head()          # extract first part of data
tail()          # extract last part of data
colnames()      # column names
rownames()      # row names
summary()
```

```
# Alter/create data:
subset()        # subset data by condition
factor()        # create grouping variable
relevel()       # change reference level
cut()           # cut numeric into intervals
round()         # rounding numbers
c()             # concatenate numerics
seq()           # create sequence
with()
aggregate()
margin.table()  # sum table entries
```

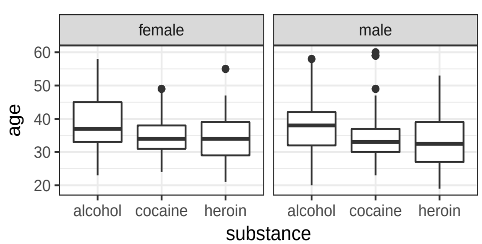
The following are examples of how some of the functions work (based on `mosaic`'s built-in data set, `HELPrct`). We assume `mosaic` is already loaded. In some chunks only the code and not the output is shown (to see the output, copy-paste the code chunk of interest into your console).

```
# Create a contingency table for 'sex' and
# 'substance':
tally(sex ~ substance, data = HELPrct)

        substance
sex      alcohol cocaine heroin
  female      36      41     30
  male       141     111     94
```

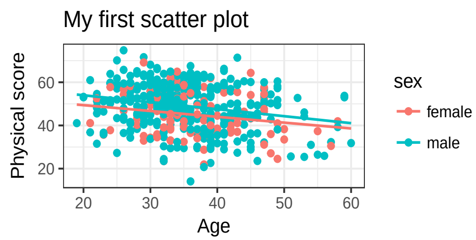
```
# Calculate mean 'age' for men and women:
mean(age ~ sex, data = HELPrct)

female   male
 36.25  35.47
```

```
# 'favstats' can be used to retrieve different
# summaries of the data (here for 'age'
# separated by sex) :
favstats(age ~ sex, data = HELPrct)

     sex min Q1 median   Q3 max  mean    sd
1 female  21 31     35 40.5  58 36.25 7.585
2   male  19 30     35 40.0  60 35.47 7.750
    n missing
1 107       0
2 346       0
```

```
# Boxplot of 'age' for each substance with
# different panels for men and women:
gf_boxplot(age ~ substance | sex, data = HELPrct)
```




```
# Make a scatterplot of 'pcs' versus 'age'
# coloured by 'sex' and add regression lines:
gf_point(pcs~age, col = ~sex, data = HELPrct) %>%
  gf_lm() %>%
  gf_labs(x = "Age",
          y = "Physical score",
          title = "My first scatter plot")
```



Note: `gf_point` creates the scatter plot, `gf_lm` adds regression lines and `gf_labs` adds a title and change axis labels.

```
# Use an exact binomial test to test whether
# the proportion of women is 50 %:
binom.test(~sex, p = 0.5, data = HELPrct)
```

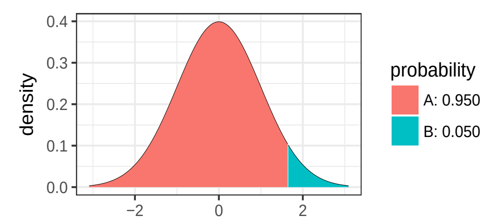
```
# Use a t-test to test whether the mean age of
# men and women are the same:
t.test(age ~ sex, data = HELPrct)
```

```
# Use a chi-square test to test for
# independence between 'homeless' and 'sex':
tab <- tally(homeless ~ sex, data = HELPrct)
chisq.test(tab)

# Use an approximate test to see whether the
# proportion of homeless is the same for men
# and women:
prop.test(homeless ~ sex, data = HELPrct)
```

```
# Investigate whether we may drop 'age' as an
# explanatory variable for 'pcs', when
# 'substance' is in the linear model too:
mod1 <- lm(pcs ~ age + substance, data = HELPrct)
mod2 <- lm(pcs ~ substance, data = HELPrct)
anova(mod1, mod2)

Analysis of Variance Table

Model 1: pcs ~ age + substance
Model 2: pcs ~ substance
  Res.Df   RSS Df Sum of Sq    F  Pr(>F)
1    449 47517
2    450 50139 -1     -2623 24.8 9.2e-07
```

Illustration of how the functions `pdist` and `qdist` works:

```
# Calculate the 95th percentile for the
# standard normal distribution (i.e., mean = 0
# and standard deviation = 1):
qdist("norm", p = 0.95, mean = 0, sd = 1)

[1] 1.645
```




```
# Calculate the probability of getting a value
# less than -1.5 for the standard normal
# distribution:
pdist("norm", q = -1.5, mean = 0, sd = 1)

[1] 0.06681
```