Stata and Paste

Edgar Treischl

2021-04-19

Introduction

Figure 1: Source: spicyexpresso2000 (https://imgur.com/gallery/IwNTmcf)

Stata and Paste? Do you need some EMS help? Fast so gut wie der Dino im Comic der Stata Helpline zeigt Dir diese Homepage wie die gängisten Stata Befehle auf Basis von Minimalbeispielen funktionieren. Der Stata Code und Output wird direkt über Stata in diese Homepage eingebunden, du kannst Dir sicher sein, dass die Minimalbeispiele funktionieren und als Basis für deine eigenen Analysen nutzen.

Alle Befehle sind mit Daten durchgeführt, die in Stata gespeichert sind. Wenn kein anderer Datensatz am Anfang des Kapitels genannt wird, dann basiert das Minimalbeispiel auf dem Auto Datensatz. Den Autodatensatz lädst du mit `sysuse auto, clear` und `describe` gibt dir eine Beschreibung des Datensatzes.

```
sysuse auto, clear
describe
```

```
Contains data from C:\PROGRA~1\Stata16\ado\base/a/auto.dta
  obs:            74                          1978 Automobile Data
```

```
 vars:            12                          13 Apr 2018 17:45
                                              (_dta has notes)
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
make            str18   %-18s                 Make and Model
price           int     %8.0gc                Price
mpg             int     %8.0g                 Mileage (mpg)
rep78           int     %8.0g                 Repair Record 1978
headroom        float   %6.1f                 Headroom (in.)
trunk           int     %8.0g                 Trunk space (cu. ft.)
weight          int     %8.0gc                Weight (lbs.)
length          int     %8.0g                 Length (in.)
turn            int     %8.0g                 Turn Circle (ft.)
displacement    int     %8.0g                 Displacement (cu. in.)
gear_ratio      float   %6.2f                 Gear Ratio
foreign         byte    %8.0g       origin    Car type
-------------------------------------------------------------------------------
Sorted by: foreign
```

Auf dieser Homepage bekommst du also immer ein Minimalbeispiel gezeigt, d.h. du siehst den Stata Code und Output. Wichtig: Rechts oben im Stata Code befindet sich Clippo. Nutze Clippo um den Code zu kopieren und passe so die Befehle für deine Analysen an. Say hi to Clippo and happy coding!

Der Comic ist natürlich nur als Scherz gedacht! Sprich uns in der Übung gezielt auf einzelne Aspekte oder Probleme an, wir werden Dir schon nicht den Kopf abreißen.

Die Homepage ist noch Work in Progress. Sprich mich bitte darauf an, wenn du Fehler findest oder wichtige Stata Befehle fehlen.

The Basics

use and *describe*

- Ein Datensatz wird mit *use* geladen
- Stata sucht im Home Directory nach dem Datensatz, wenn keine Angabe zum Pfad gemacht wurde
- Die Option *clear* löscht alle bis dato geladenen Datensätze
- Der Befehl *describe* beschreibt die Daten und zeigt Dir welche Variablen in einem Datensatz enthalten sind

```
use "https://www.stata-press.com/data/r16/titanic800", clear
describe
```

```
(Titanic passenger survival (Extract))


Contains data from https://www.stata-press.com/data/r16/titanic800.dta
  obs:           800                          Titanic passenger survival
                                                (Extract)
 vars:             4                          22 Feb 2019 13:24
                                               (_dta has notes)
------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
------------------------------------------------------------------------------
class           byte    %9.0g      class      Class
adult           byte    %9.0g      age        Adult
male            byte    %9.0g      sex        Male
survived        byte    %9.0g      survived   Survived
------------------------------------------------------------------------------
Sorted by:
```

(sys)use

- Mit *use* greifst du auf Daten zu, die lokal gespeichert sind
- Mit *sysuse* greifst du auf Daten zu, die in Stata gespeichert sind
- Dies ist praktisch für Illustrationszwecke, alle Beispiel hier greifen auf solche Daten zurück

```
sysuse auto, clear
des
```

```
(1978 Automobile Data)


Contains data from C:\PROGRA~1\Stata16\ado\base/a/auto.dta
  obs:            74                          1978 Automobile Data
 vars:            12                          13 Apr 2018 17:45
                                              (_dta has notes)
------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
------------------------------------------------------------------------------
make            str18   %-18s                 Make and Model
```

```
price           int     %8.0gc                Price
mpg             int     %8.0g                 Mileage (mpg)
rep78           int     %8.0g                 Repair Record 1978
headroom        float   %6.1f                 Headroom (in.)
trunk           int     %8.0g                 Trunk space (cu. ft.)
weight          int     %8.0gc                Weight (lbs.)
length          int     %8.0g                 Length (in.)
turn            int     %8.0g                 Turn Circle (ft.)
displacement    int     %8.0g                 Displacement (cu. in.)
gear_ratio      float   %6.2f                 Gear Ratio
foreign         byte    %8.0g      origin     Car type
-------------------------------------------------------------------------------
Sorted by: foreign
```

list

- Mit dem Befehl *list* kann der Datensatz im Ausgabefenster angezeigt werden. Im nächsten Output Fenster werden die ersten fünf Fälle in Tabellenform angezeigt
- Ohne Angabe der Variablen wird der ganze Datensatz ausgegeben
- List ist praktisch um zu überprüfen, ob die Datenaufbereitung geklappt hat

```
list make - headroom in 1/5, table
```

```
. list make - headroom in 1/5, table

     +-------------------------------------------------+
     | make            price   mpg   rep78   headroom |
     |-------------------------------------------------|
  1. | AMC Concord     4,099    22       3        2.5 |
  2. | AMC Pacer       4,749    17       3        3.0 |
  3. | AMC Spirit      3,799    22       .        3.0 |
  4. | Buick Century   4,816    20       3        4.5 |
  5. | Buick Electra   7,827    15       4        4.0 |
     +-------------------------------------------------+
```

order

- Mit dem Befehl *order* lässt sich der Datensatz sortieren
- Einzelne oder mehrere Variablen können jeweils vor oder nach einer anderen Variablen sortiert werden

```
*Sortiere die Variable price Vor die Variable make
order price, before(make)
list price - headroom in 1/4, table
```

```
 . *Sortiere die Variable price Vor die Variable make
. order price, before(make)

. list price - headroom in 1/4, table

     +-------------------------------------------------+
     | price   make            mpg   rep78   headroom |
     |-------------------------------------------------|
  1. | 4,099   AMC Concord      22       3        2.5 |
  2. | 4,749   AMC Pacer        17       3        3.0 |
  3. | 3,799   AMC Spirit       22       .        3.0 |
  4. | 4,816   Buick Century    20       3        4.5 |
```

```
     +-------------------------------------------------+
*Oder danach
order price, after(make)
list make - headroom in 1/4, table

 . *Oder danach
. order price, after(make)

. list make - headroom in 1/4, table

     +-------------------------------------------------+
     | make            price   mpg   rep78   headroom |
     |-------------------------------------------------|
  1. | AMC Concord     4,099    22       3        2.5 |
  2. | AMC Pacer       4,749    17       3        3.0 |
  3. | AMC Spirit      3,799    22       .        3.0 |
  4. | Buick Century   4,816    20       3        4.5 |
     +-------------------------------------------------+
```

tab

- Der Befehl *tab* gibt eine einfache Tabelle aus
- Folgen zwei Variablennamen im Befehl, erhält man eine Kreuztabelle
- Mit *tab* können auch Dummy Variablen einfach erstellt werden

```
tab foreign

 . tab foreign

   Car type |      Freq.     Percent        Cum.
------------+-----------------------------------
   Domestic |         52       70.27       70.27
    Foreign |         22       29.73      100.00
------------+-----------------------------------
      Total |         74      100.00
```

- *tab var, gen(new_var_)* erstellt n Dummy Variable new_var_(1 bis n) gemäß der Ausprägungen der Variable var

```
tab rep78, gen(rep78_dummy_)
list rep78_dummy_1 - rep78_dummy_5 in 1/5

 . tab rep78, gen(rep78_dummy_)

     Repair |
Record 1978 |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |          2        2.90        2.90
          2 |          8       11.59       14.49
          3 |         30       43.48       57.97
          4 |         18       26.09       84.06
          5 |         11       15.94      100.00
------------+-----------------------------------
      Total |         69      100.00

. list rep78_dummy_1 - rep78_dummy_5 in 1/5
```

```
     +-----------------------------------------------------+
     | rep78_~1   rep78_~2   rep78_~3   rep78_~4   rep78_~5 |
     |-----------------------------------------------------|
  1. |        0          0          1          0          0 |
  2. |        0          0          1          0          0 |
  3. |        .          .          .          .          . |
  4. |        0          0          1          0          0 |
  5. |        0          0          0          1          0 |
     +-----------------------------------------------------+
```

Ados installieren

- In Stata können Erweiterungen (Ados) installiert werden
- *ssc install* installiert das entsprechenden Package
- *findit* sucht nach dem entsprechenden Package, beispielweise wenn man nur einen Befehl kennt

```
ssc install ado_name, replace
findit command
```

Pfad angeben/finden

- *cd* gibt den Pfad (Speicherort) an, an dem nach Daten gesucht werden oder Abbildungen exportiert werden

```
cd "C:\path"
use data.dta
```

display, keep, save, edit

- Stata als Taschenrechner nutzen

```
display 1 + 3
```

- Daten eingrenzen und löschen

```
keep varlist (if) /// löscht die nicht (!) angegebenen Variable(n), wenn die if Bedingung erfüllt ist
drop varlist (if) /// löscht Variable(n) (wenn if Bedingung erfüllt ist)
```

- *Save* speichert einen neuen Datensatz, *replace* ersetzt bereits vorhandenen Datensatz (sei vorsichtig dabei!)

```
save "Pfad\meine_bearbeiteter_datensatz.dta", replace
```

- Aufrufen des Data Editors durch *edit*

```
edit
```

Data Preparation

recode

- Mit *recode* werden die Ausprägungen einer Variable rekodiert, beispielsweise um die Ausprägungen einer Variable zusammenzufassen
- Vorsicht: Mit *recode* wird die Ausgangsvariable rekodiert! Mit der Option *generate(new_variable)* wird zusätzlich eine neue Variable generiert und die Ausgangsvariable bleibt unversehrt.
- Zum Rekodieren kann die Option *nolab* genutzt werden, diese zeigt Ausprägungen anstelle von Labels an.

- Im Stata Output werden die Werte der Variable *foreign* von 0 auf 1; bzw von 1 auf 0 rekodiert, beispielsweise um eine leichtere Interpretation der Dummy Variable zu erhalten.

```
sysuse auto, clear
tab foreign
tab foreign, nolab
recode foreign (0 = 1) (1 = 0), gen(foreign_recode)
tab2 foreign foreign_recode

(1978 Automobile Data)


   Car type |      Freq.     Percent        Cum.
------------+-----------------------------------
   Domestic |         52       70.27       70.27
    Foreign |         22       29.73      100.00
------------+-----------------------------------
      Total |         74      100.00


   Car type |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |         52       70.27       70.27
          1 |         22       29.73      100.00
------------+-----------------------------------
      Total |         74      100.00

(74 differences between foreign and foreign_recode)


-> tabulation of foreign by foreign_recode

           |   RECODE of foreign
           |      (Car type)
  Car type |         0          1 |     Total
-----------+----------------------+----------
  Domestic |         0         52 |        52
   Foreign |        22          0 |        22
-----------+----------------------+----------
     Total |        22         52 |        74
```

- Bitte bei der Erstellung oder Rekodierung einer Variable stets nach der Datenaufbereitung prüfen ob dies fehlerfrei funktioniert hat (bspw. über *tab Befehl*)

generate und *replace*

- Mit *generate* wird eine neue Variable erstellt und je nach Angabe werden spezifische Werte (auch missings) vergeben
- Mit *replace* können diese Werte ausgetauscht werdem, also beispielsweise nach der Ausprägung einer zu recodierenden Variablen ersetzt werden
- Beispiel: Die Variable foreign zeigt an, ob ein Auto ausländisch (1) ist
- Mit *generate* und *replace* können wir eine neue Variable (domestic) erzeugen die auf 1 springt, wenn ein Auto ausländisch ist anstelle der alten Codierung

```
gen domestic = 1
replace domestic = 0 if foreign == 1
```

```
tab2 foreign domestic

(22 real changes made)


-> tabulation of foreign by domestic

           |       domestic
  Car type |         0          1 |     Total
-----------+----------------------+----------
  Domestic |         0         52 |        52
   Foreign |        22          0 |        22
-----------+----------------------+----------
     Total |        22         52 |        74
```

egen

- *egen* bedeutet: Extensions to generate
- Beispielsweise lässt sich so der Mean einer Variable berechnen und dieser kann direkt als neue Variable gespeichert werden

```
egen mean_price = mean(price)
list mean_price in 1/4, table

 . egen mean_price = mean(price)

. list mean_price in 1/4, table

     +----------+
     | mean_p~e |
     |----------|
  1. | 6165.257 |
  2. | 6165.257 |
  3. | 6165.257 |
  4. | 6165.257 |
     +----------+
```

- Zur Generierung von Summary statistics können eine Reihe an Funktionen: count(), iqr(), kurt(), mad(), max(), mdev(), mean(), median(), min(), mode(), pc(), pctile(), sd(), skew(), and total() genutzt werden.
- Häufig muss der Datensatz noch entsprechend sortiert werden. Beispielweise bei einer zeitlicher Struktur und mehrere Wellen in den Daten. Dies zeigt der folgende Output exemplarisch:

```
by sortier_var, sort: egen new_var = median(var)
```

label

- Your variables need some nice labels?
- Der Befehl *lab var* gibt der Variable ein Label
- Der Befehl *lab def* definiert die Labels für die Ausprägungen der Variable
- Der Befehl *lab val* fügt die definierten Labels der Variablen hinzu

```
*Einer Variable einen Namen, Label geben
label variable domestic "Inländische Autos"
```

```
*Ausprägungen müssen definiert werden und können so für mehrere Variablen genutzt werden
label define car_label 0 "Ausländische Autos" 1 "Inländische Autos"

*Ausprägungen müssen der Variable zugeordnet werden
label values domestic car_label

*Did it work?
tab domestic
```

```
 Inländische Autos |      Freq.     Percent        Cum.
-------------------+-----------------------------------
Ausländische Autos |         22       29.73       29.73
 Inländische Autos |         52       70.27      100.00
-------------------+-----------------------------------
             Total |         74      100.00
```

Mathematische Transformationen

Hier eine erste kurze Sammlung zentraler mathematischer Transformationen in Stata:

- Logarithmus von Variable(n) erstellen
- Polynome
- Variable am Mean zentrieren

```
gen ln_age = ln(age)         // Logarithmus
gen quad_age = (age*age)     // Quadratischer Term
center age , gen (zent_age) // Variable zentrieren
```

Linear Regression

regress

- Hat das Gewicht eines Autos (weight) einen Einfluss auf den Verbrauch/Reichweite (mpg miles pro galion) des Autos?
- Mit *regress Y X* berechnest du den Einfluss einer metrischen oder binären Dummy Variable X auf eine metrischen Zielvariable Y

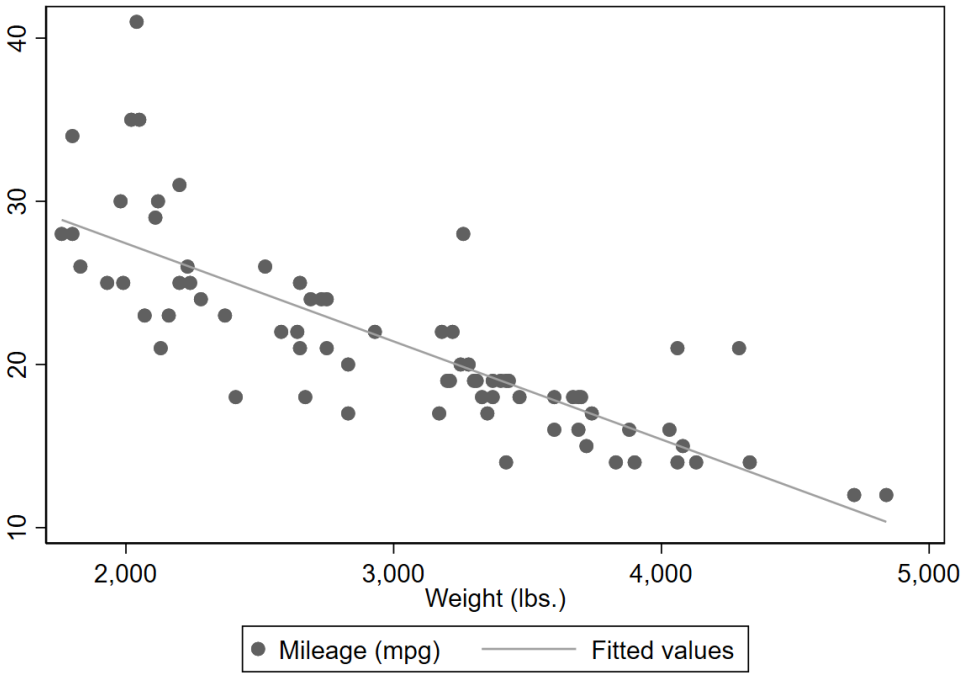
```
sysuse auto, clear
regress mpg weight

(1978 Automobile Data)
```

```
      Source |       SS           df       MS      Number of obs   =        74
-------------+----------------------------------   F(1, 72)        =    134.62
       Model |   1591.9902         1   1591.9902   Prob > F        =    0.0000
    Residual |  851.469256        72  11.8259619   R-squared       =    0.6515
-------------+----------------------------------   Adj R-squared   =    0.6467
       Total |  2443.45946        73  33.4720474   Root MSE        =    3.4389

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |  -.0060087   .0005179   -11.60   0.000    -.0070411   -.0049763
       _cons |   39.44028   1.614003    24.44   0.000     36.22283    42.65774
------------------------------------------------------------------------------
```

Interpretation: Erhöht sich das Gewicht eines Autos um eine Einheit, dann verringert sich die Reichweite im Durchschnitt um -.006 Einheiten (Miles pro galion). Der Effekt ist statistisch signifikant.

- By the way. Inwiefern die Modellannahmen gegeben sind, wurden bei keinem der hier aufgeführten Beispiele geprüft. Das ist Dein Job ;)! Beispielsweise sollte man prüfen, inwiefern überhaupt von einem linearen Zusammenhang ausgegangen werden kann. Im Scatter ist dies meistens gut sichtbar:

```
twoway scatter mpg weight || lfit mpg weight
```



Vorhersagen per Hand

- Wie hoch fällt die vorhergesagte Laufleistung bei einem durchschnittlichen Gewicht von 3019 Pfund aus?
- *coeflegend* zeigt Dir, wie die Terme der Regression heißen

```
quietly regress mpg weight
regress, coeflegend
```

```
      Source |       SS           df       MS      Number of obs   =        74
-------------+----------------------------------   F(1, 72)        =    134.62
       Model |   1591.9902         1   1591.9902   Prob > F        =    0.0000
    Residual |  851.469256        72  11.8259619   R-squared       =    0.6515
-------------+----------------------------------   Adj R-squared   =    0.6467
       Total |  2443.45946        73  33.4720474   Root MSE        =    3.4389

------------------------------------------------------------------------------
         mpg |      Coef.  Legend
-------------+----------------------------------------------------------------
      weight |  -.0060087  _b[weight]
       _cons |   39.44028  _b[_cons]
------------------------------------------------------------------------------
```

- Mit *display* können wir dann eine Vorhersage per Hand berechnen:

```
display _b[_cons] + _b[weight] * 3019

21.300058
```

esttab

- *esttab* ist ein ado und kann für den Modellvergleich genutzt werden, nachdem das Ado installiert wurde
- Hierfür müssen wir die Modelle mit *estimates store* zuvor speichern
- Der Befehl *quietly* unterdrückt den Output der eigentlichen Regressionstabelle
- *esttab* kann dann genutzt werden, um die Modelle miteinander zu vergleichen

```
quietly regress mpg weight
estimates store model1
quietly regress mpg weight headroom trunk length turn
estimates store model2

esttab model1 model2
```

```
                      (1)             (2)
                      mpg             mpg
--------------------------------------------
weight           -0.00601***     -0.00375*
                 (-11.60)         (-2.24)

headroom                          0.0121
                                  (0.02)

trunk                            -0.0367
                                 (-0.23)

length                           -0.0675
                                 (-1.05)

turn                             -0.0627
                                 (-0.33)

_cons                39.44***        48.25***
                    (24.44)          (6.81)
--------------------------------------------
N                       74              74
--------------------------------------------
t statistics in parentheses
* p<0.05, ** p<0.01, *** p<0.001
```

coefplot

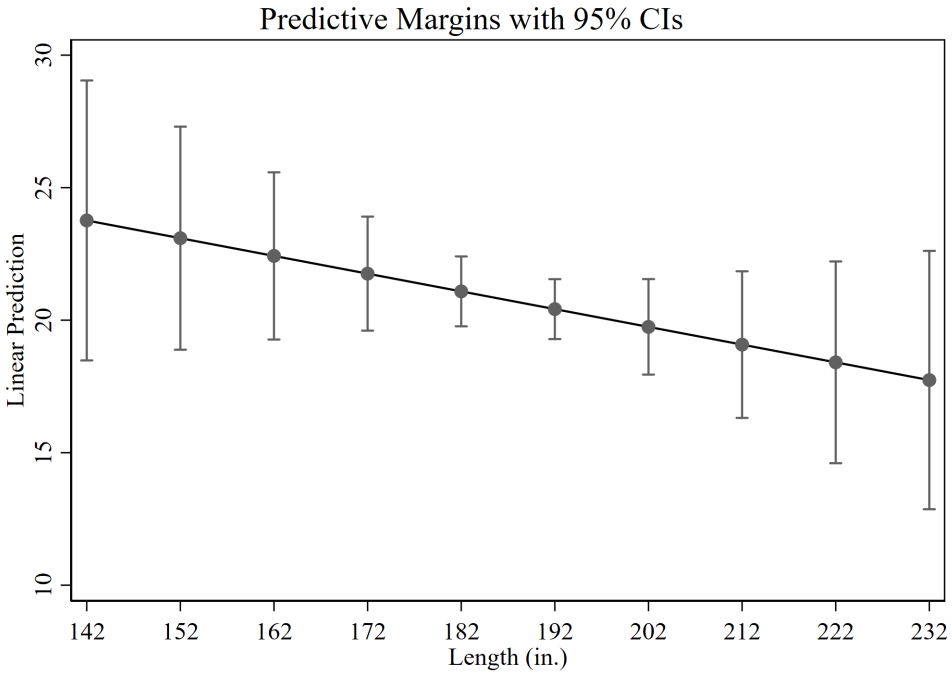
- Nutze *coefplot* zur Visualisierung von Regressionsergebnisse. Coefplot plottet die Koeffizienten der Regression
- Mehr Infos zu Coefplot?

Interaktionen (##)

- Interaktionsterme werden durch ## ausgegeben und Visualisierrung der Interaktionen erhällt man durch den *margins* Befehl
- Im nächsten Output: Interaktion zweier metrischer (`c.`) Variablen:

- Ferner muss bei metrischen Variablen die Range der Vorhersage angeben werden. Hier: `length=(142(10)233)` bestimmt die Range der vorhergesagten Werte, mit dem Minimum (142), dem Maximum (233) und sinnvolle Zwischenschritte, die zuvor für die `length` Variable berechnet wurden

```
regress mpg c.weight##c.length
quietly margins, at(length=(142(10)233))
marginsplot
```



Interpretation: Die Interaktion dieser beiden Variablen ist nicht signifikant, die Konfidenzintervalle überlappen sich deutlich.

- Alternativ: Bei Interaktionen zwei binärer `(i)` Variablen:

```
regress Y i.X##Z
margins i.X##i.Z
marginsplot
```

Log Modelle: Lin-Log

- UV wird logarithmiert

```
gen ln_weight = ln(weight)
regress mpg ln_weight
```

```
      Source |       SS           df       MS      Number of obs   =        74
-------------+----------------------------------   F(1, 72)        =    147.33
       Model |  1641.35428         1  1641.35428   Prob > F        =    0.0000
    Residual |  802.105178        72  11.1403497   R-squared       =    0.6717
-------------+----------------------------------   Adj R-squared   =    0.6672
       Total |  2443.45946        73  33.4720474   Root MSE        =    3.3377

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
```

```
-------------+----------------------------------------------------------------
   ln_weight |  -17.80317   1.466715   -12.14   0.000    -20.72702   -14.87933
       _cons |   163.3444   11.70898    13.95   0.000      140.003    186.6858
------------------------------------------------------------------------------
```

Interpretation: Mit einem Anstieg des Gewichts um 1 Prozent reduziert sich die durchschnittliche Reichweite um -.178 (beta/100) Einheiten.

Log-Lin

- AV wird logarithmiert

```
gen ln_mpg = ln(mpg)
regress ln_mpg weight
```

```
      Source |       SS           df       MS      Number of obs   =        74
-------------+----------------------------------   F(1, 72)        =    179.26
       Model |  3.52530845         1  3.52530845   Prob > F        =    0.0000
    Residual |   1.4159149        72  .019665485   R-squared       =    0.7134
-------------+----------------------------------   Adj R-squared   =    0.7095
       Total |  4.94122335        73  .067687991   Root MSE        =    .14023

------------------------------------------------------------------------------
      ln_mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |  -.0002828   .0000211   -13.39   0.000    -.0003249   -.0002407
       _cons |   3.878298   .0658171    58.93   0.000     3.747094    4.009502
------------------------------------------------------------------------------
```

Interpretation: Erhört sich das Gewicht um eine Einheit, so reduziert sich die Reichweite im Durchschnitt approximativ um 0,02 %.

Log-Log

- UV und AV sind logarithmiert

```
regress ln_mpg ln_weight
```

```
      Source |       SS           df       MS      Number of obs   =        74
-------------+----------------------------------   F(1, 72)        =    179.41
       Model |  3.52612925         1  3.52612925   Prob > F        =    0.0000
    Residual |   1.4150941        72  .019654085   R-squared       =    0.7136
-------------+----------------------------------   Adj R-squared   =    0.7096
       Total |  4.94122335        73  .067687991   Root MSE        =    .14019

------------------------------------------------------------------------------
      ln_mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
   ln_weight |  -.8251737    .061606   -13.39   0.000    -.9479829   -.7023645
       _cons |   9.608391   .4918087    19.54   0.000     8.627989    10.58879
------------------------------------------------------------------------------
```

Interpretation: Erhöht sich das Gewicht um 1 %, reduziert sich die Reichweite im Durchschnitt um 8,2 %.

Optionen: robust

- Robuste Standardfehler erhällt man durch die Option `vce(robust)`

```
regress mpg weight, vce(robust)

Linear regression                               Number of obs     =         74
                                                F(1, 72)          =     105.83
                                                Prob > F          =     0.0000
                                                R-squared         =     0.6515
                                                Root MSE          =     3.4389

------------------------------------------------------------------------------
             |               Robust
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |  -.0060087   .0005841   -10.29   0.000     -.007173   -.0048443
       _cons |   39.44028    1.98832    19.84   0.000     35.47664    43.40393
------------------------------------------------------------------------------
```

e(sample)

- *e(sample)* markiert, welche Fälle in der Analyse berücksichtigt wurden, so kann die Fallzahl beim Vergleich mehrerer Modelle "konstant" gehalten werden
- Hierfür wird zuerst das umfangreichere Modell gerechnet, da jede weitere Variable im Model die Anzahl an Missing erhöhen kann. Danach wird eine Variable mit dem Namen `sample` generiert die anzeigt, welche Fälle berücksichtigt wurden
- Auf Basis dieser Variable können dann die "kleineren" Modelle mit Hilfe des *if* berechnet werden

```
quietly regress mpg weight headroom trunk length turn
gen sample = 1 if e(sample)

quietly regress mpg weight if sample == 1
quietly regress mpg weight headroom trunk length turn if sample == 1
```

ereturn list

- *ereturn list* zeigt, welche Terme Stata bei der Regression berechnet hat und wie die einzelnen Terme intern heißen

```
ereturn list

scalars:
                  e(N) =  74
              e(df_m) =  5
              e(df_r) =  68
                 e(F) =  26.66054023704533
                e(r2) =  .6622002606043995
              e(rmse) =  3.4839969622233
               e(mss) =  1618.059490830339
               e(rss) =  825.3999686291203
              e(r2_a) =  .63736204447237
                e(ll) =  -194.2381619431207
              e(ll_0) =  -234.3943376482347
              e(rank) =  6

macros:
            e(cmdline) : "regress mpg weight headroom trunk length turn if .."
```

```
              e(title) : "Linear regression"
           e(marginsok) : "XB default"
                 e(vce) : "ols"
               e(depvar) : "mpg"
                 e(cmd) : "regress"
          e(properties) : "b V"
             e(predict) : "regres_p"
               e(model) : "ols"
            e(estat_cmd) : "regress_estat"

matrices:
                  e(b) :  1 x 6
                  e(V) :  6 x 6

functions:
              e(sample)
```

Logistische Regression

- Mit einer logistischen Regression schätzt Du den Effekt einer metrischen Variable (oder einer Dummy Variable) auf ein binäres Outcome
- Unser Beispiel: Hat das Geschlecht einen Effekt, ob eine Person das Titanic Unglück überlebt hat?

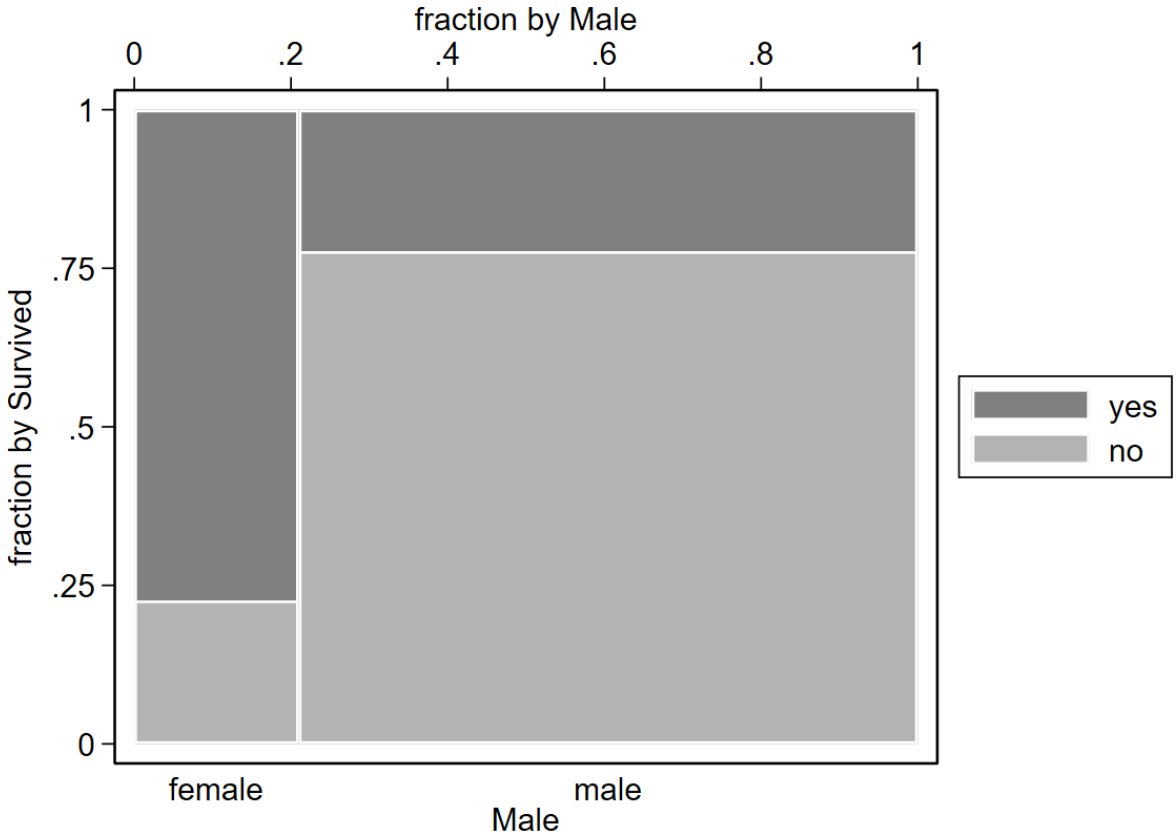
Mit dem Titanic Datensatz wird im folgenden die Grundbefehle einer logistischen Regression in Stata gezeigt:

```
use "https://www.stata-press.com/data/r16/titanic800", clear
des
```

```
(Titanic passenger survival (Extract))


Contains data from https://www.stata-press.com/data/r16/titanic800.dta
  obs:           800                          Titanic passenger survival
                                                (Extract)
 vars:             4                          22 Feb 2019 13:24
                                               (_dta has notes)
------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
------------------------------------------------------------------------------
class           byte    %9.0g      class      Class
adult           byte    %9.0g      age        Adult
male            byte    %9.0g      sex        Male
survived        byte    %9.0g      survived   Survived
------------------------------------------------------------------------------
Sorted by:
```

Wir schätzen den Effekt des Geschlechts (male) auf die Überlebenchance (survived) einer Person. Neben einer einfachen Kreuztabelle zeigt beispielsweise auch ein *spineplot* welcher Anteil an Männern und Frauen den Titanic Unfall (nicht) überlebt hat.

```
*spineplot ist im ado tabplot und muss installiert werden
spineplot survived male // Splineplot
```



Mehr als drei Viertel der weiblichen Passagiere hat die Titanic überlebt; währrend ein deutlich größerer Anteil der männlichen Passagiere das Titanic Unglück nicht überlebt hat. Wie viel größer ist das Risiko (bzw. Chancenverhältnis) von Männern gegenüber Frauen? Let's run a logistic regression to find an answer to this question.

logit

- Mit dem *logit* Befehl werden in Stata die logarithmierten Chancen berechnet. Dies ist schwierig zu interpretieren
- Wir können zumindest die Vorzeichen und die Signifikanz interpretieren

```
logit survived male

Iteration 0:   log likelihood =  -512.1623
Iteration 1:   log likelihood = -425.53116
Iteration 2:   log likelihood = -425.28715
Iteration 3:   log likelihood = -425.28708
Iteration 4:   log likelihood = -425.28708

Logistic regression                             Number of obs     =        800
                                                LR chi2(1)        =     173.75
                                                Prob > chi2       =     0.0000
Log likelihood = -425.28708                     Pseudo R2         =     0.1696

------------------------------------------------------------------------------
    survived |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
```

```
-------------+----------------------------------------------------------------
        male |  -2.477633   .2076944   -11.93   0.000    -2.884706   -2.070559
       _cons |   1.229948   .1844128     6.67   0.000     .8685058    1.591391
------------------------------------------------------------------------------
```

Interpretation: Männer haben im Durchschnitt einen 2.477 geringere Log-Odds als Frauen. Der Effekt ist signifikant.

or

- Etwas leichter ist die Interpretation eines Odds Ratios (OR) - das Chancenverhältnis
- Hierfür müssen wir die Exponentialfunktion anwenden

```
di exp(_b[male])
```

```
.08394172
```

- In Stata erhällt man den OR mit der Option *or*:

```
logit survived male, or
```

```
Iteration 0:   log likelihood =  -512.1623
Iteration 1:   log likelihood = -425.53116
Iteration 2:   log likelihood = -425.28715
Iteration 3:   log likelihood = -425.28708
Iteration 4:   log likelihood = -425.28708

Logistic regression                             Number of obs     =        800
                                                LR chi2(1)        =     173.75
                                                Prob > chi2       =     0.0000
Log likelihood = -425.28708                     Pseudo R2         =     0.1696

------------------------------------------------------------------------------
    survived | Odds Ratio   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        male |   .0839417   .0174342   -11.93   0.000     .0558712    .1261153
       _cons |   3.421053    .630886     6.67   0.000     2.383347    4.910574
------------------------------------------------------------------------------
Note: _cons estimates baseline odds.
```

Interpretation: Männer haben im Vergleich zu Frauen durchschnittlich eine um den Faktor .083 geringe Chance die Titanic zu überleben. Der Effekt ist statistisch signifikant.

OR per Kreutabelle berechnen

- Was der OR wirklich inhaltlich aussagt, lässt sich ganz gut nachvollziehen, wenn wir den OR auf Basis einer Kreuztabelle ausrechnem

```
*Odds Ratio per Hand ausrechnen
tab2 survived male
```

```
-> tabulation of survived by male

           |         Male
  Survived |    female       male |     Total
-----------+----------------------+----------
        no |        38        491 |       529
       yes |       130        141 |       271
```

```
-----------+----------------------+----------
     Total |       168        632 |       800
```

- Bei den Männer haben 141 überlebt, 491 aber nicht. Wenn wir diese zwei Zahlen ins Verhältnis zueinander setzen, erhalten wir die Odds. Die Odds zu Überleben von Männer ist 141/491.
- Analog müssen wir das gleiche für die Frauen im Datensatz machen.
- Zum Glück können wir Stata als Taschenrechner mit *display* nutzen!

```
*Odds der Männer
display 141/491

*Odds der Frauen
display 130/38

.28716904

3.4210526
```

- Und nun der OR, also das Chancenverhältnis der Männer gegenüber den Frauen die Titanic zu überleben. Wir teilen also die Odds der Männer durch den Wert der Frauen:

```
*OR
display .28716904/3.4210526

.08394172
```

Und kommen so zum gleichen Ergebnis wie die logistische Regression.

margins

- Die Ergebnisse einer logistischen Regression sind NICHT als Wahrscheinlichkeit interpretierbar, jedoch können wir Wahrscheinlichkeiten berechnen mit Hilfe des *margins* Befehls
- Marginaleffekte am Mittelwertsvektor (Befehl: margins, dydx(*) atmeans)
- Oder: Average Marginal Effects

```
quietly logit survived male
margins, dydx(*) continuous

Average marginal effects                        Number of obs     =        800
Model VCE    : OIM

Expression   : Pr(survived), predict()
dy/dx w.r.t. : male

------------------------------------------------------------------------------
             |            Delta-method
             |      dy/dx   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        male |   -.430326   .0228731   -18.81   0.000    -.4751565   -.3854955
------------------------------------------------------------------------------
```

Interpretation: Männer haben durchschnittliche eine 43 Prozentpunkte geringere Wahrscheinlichkeit die Titanic zu überleben.

lrtest

- LR Test dient zu Modellvergleiche
- Die Nullhypothese besagt, dass sich die Likelihoods von Modell 1 und Modell 2 nicht unterscheiden.

```
quietly logit survived male
est store m1
quietly logit survived male adult
est store m2

lrtest m1 m2

Likelihood-ratio test                                 LR chi2(1)  =      2.20
(Assumption: m1 nested in m2)                         Prob > chi2 =    0.1381
```

- Der p-Wert zeigt, dass es keinen signifikanten Unterschied in der Erklärungskraft von Modell 1 und 2 gibt.

fitstat

- *fitstat* gibt eine Variation an Gütemaße aus (*fitstat* ist ein Ado und muss installiert werden)
- McFaddens Pseudo-R-Quadrat: Keine analoge Interpretation wie R^2, aber höhere Werte deuten auf ein besseres Modell hin
- AIC und BIC: Kleinere Werte deuten auf ein besseres Modell hin
- Wir können *fitstat* direkt nach der Regression rufen oder der Beispielout zeigt, wie wir zuerst die Gütemaße für Model 1 berechnen, diese speichern und mit den Gütemaßen von Modell 2 vergleichen (*fitstat, using(m1)*)

```
quietly logit survived male
quietly fitstat, saving(m1)
quietly logit survived male adult
fitstat, using(m1)

Measures of Fit for logit of survived

                           Current            Saved       Difference
Model:                       logit            logit
N:                             800              800                0
Log-Lik Intercept Only:   -512.162         -512.162            0.000
Log-Lik Full Model:       -424.187         -425.287            1.100
D:                         848.375(797)     850.574(798)     -2.200(-1)
LR:                        175.950(2)       173.750(1)        2.200(1)
Prob > LR:                   0.000            0.000            0.000
McFadden's R2:               0.172            0.170            0.002
McFadden's Adj R2:           0.166            0.166            0.000
Maximum Likelihood R2:       0.197            0.195            0.002
Cragg & Uhler's R2:          0.273            0.270            0.003
McKelvey and Zavoina's R2:   0.241            0.237            0.004
Efron's R2:                  0.228            0.225            0.003
Variance of y*:              4.333            4.310            0.023
Variance of error:           3.290            3.290            0.000
Count R2:                    0.776            0.776            0.000
Adj Count R2:                0.339            0.339            0.000
AIC:                         1.068            1.068           -0.000
AIC*n:                     854.375          854.574           -0.200
BIC:                     -4479.261        -4483.746            4.485
BIC':                     -162.581         -167.066            4.485

Difference of    4.485 in BIC' provides positive support for saved model.
```

Hier scheint auch das Model 1 minimal besser an die Daten angepasst zu sein, da zumindest das BIC kleiner

ausfällt.

classification

- Anhand einer Klassifikationstabelle, kann die korrekte Vorhersage, die Sensitivität und die Spezifität abgelesen werden.

```
estat classification
```

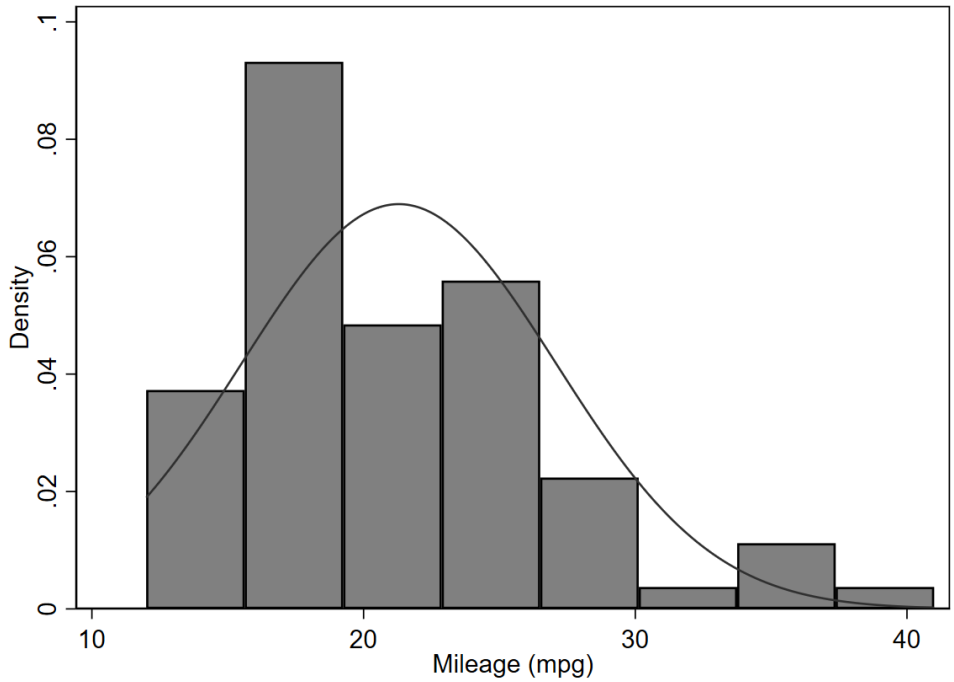
```
Logistic model for survived

              -------- True --------
Classified |         D            ~D  |      Total
-----------+--------------------------+-----------
     +     |       130            38  |        168
     -     |       141           491  |        632
-----------+--------------------------+-----------
   Total   |       271           529  |        800

Classified + if predicted Pr(D) >= .5
True D defined as survived != 0
--------------------------------------------------
Sensitivity                     Pr( +| D)   47.97%
Specificity                     Pr( -|~D)   92.82%
Positive predictive value       Pr( D| +)   77.38%
Negative predictive value       Pr(~D| -)   77.69%
--------------------------------------------------
False + rate for true ~D        Pr( +|~D)    7.18%
False - rate for true D         Pr( -| D)   52.03%
False + rate for classified +   Pr(~D| +)   22.62%
False - rate for classified -   Pr( D| -)   22.31%
--------------------------------------------------
Correctly classified                        77.63%
--------------------------------------------------
```

Interpretation: In unserem Fall wurden 77.63% der Fälle richtig klassifiziert. Die Sensitiviät liegt bei 47.97%, d.h. 47.97% der Personen die als überlebend identifiziert wurden, haben auch tatsächlich überlebt. Und die Spezifität: 92.82% der Personen die als nicht-überlebend identifiziert wurden, haben auch tatsächlich nicht überlebt.

Visualisierungen

Histogramm

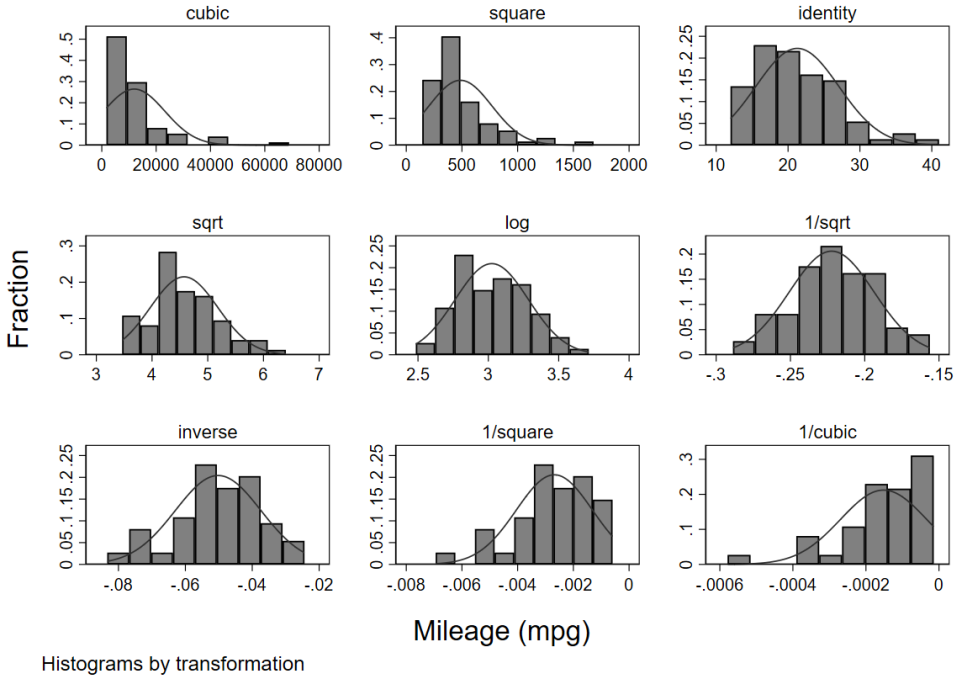
```
*Histogram inkl. Normalverteilung
sysuse auto, clear
hist mpg, norm
```



Gladder

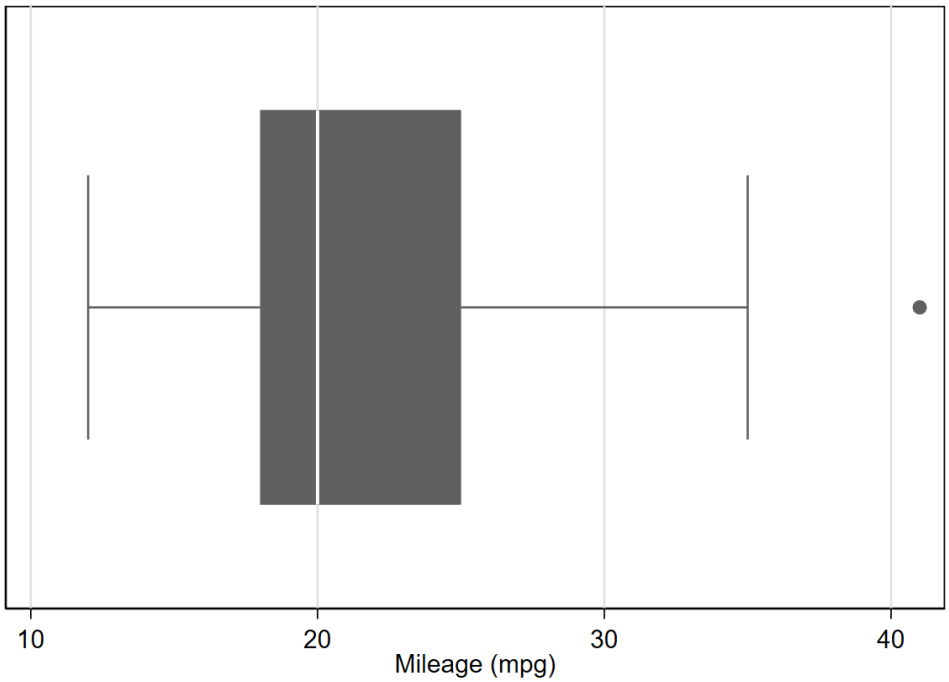
- *Gladder* transfomiert eine numerische Variable und zeigt dir wie die Verteilung nach der Transformation aussieht

```
gladder mpg, fraction
```

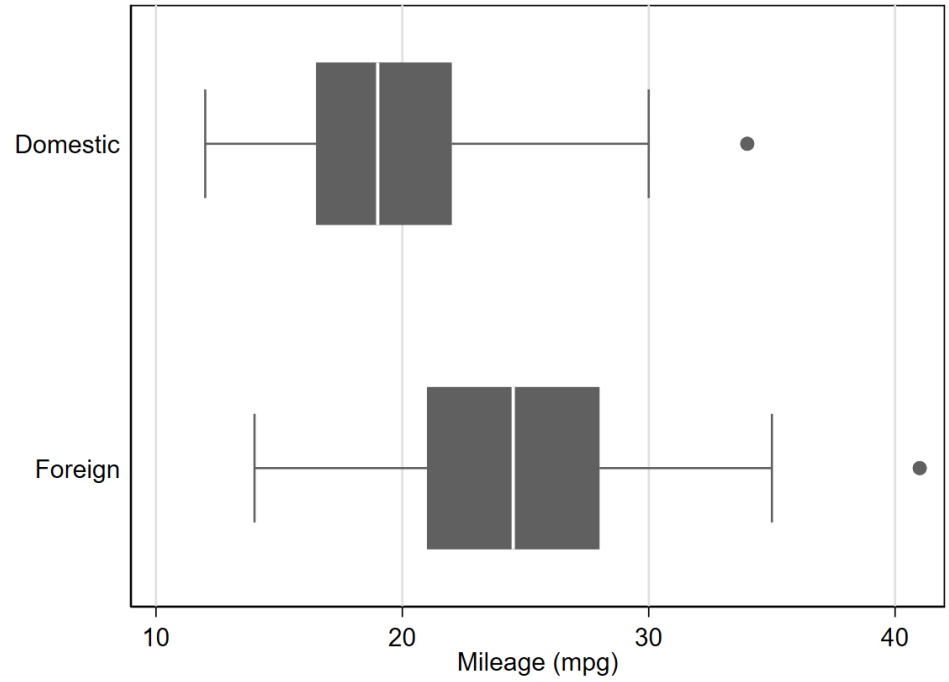


Box-Plot

```
*Box-Plot
graph box mpg
```



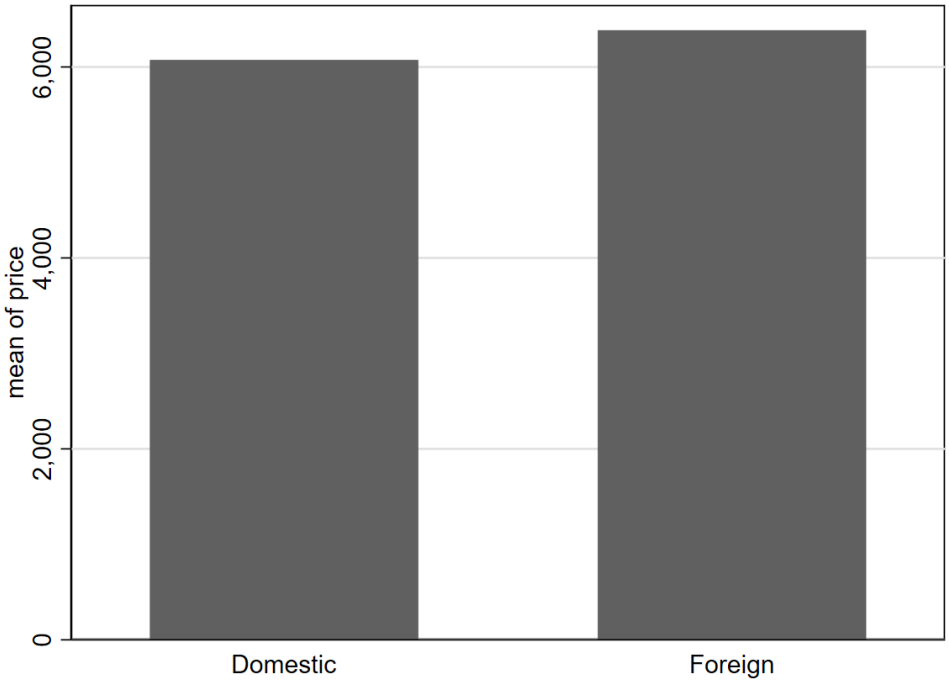

```
*Horizontal und für mehrere Gruppen
graph hbox mpg, over (foreign)
```



Deskription

Bar graph

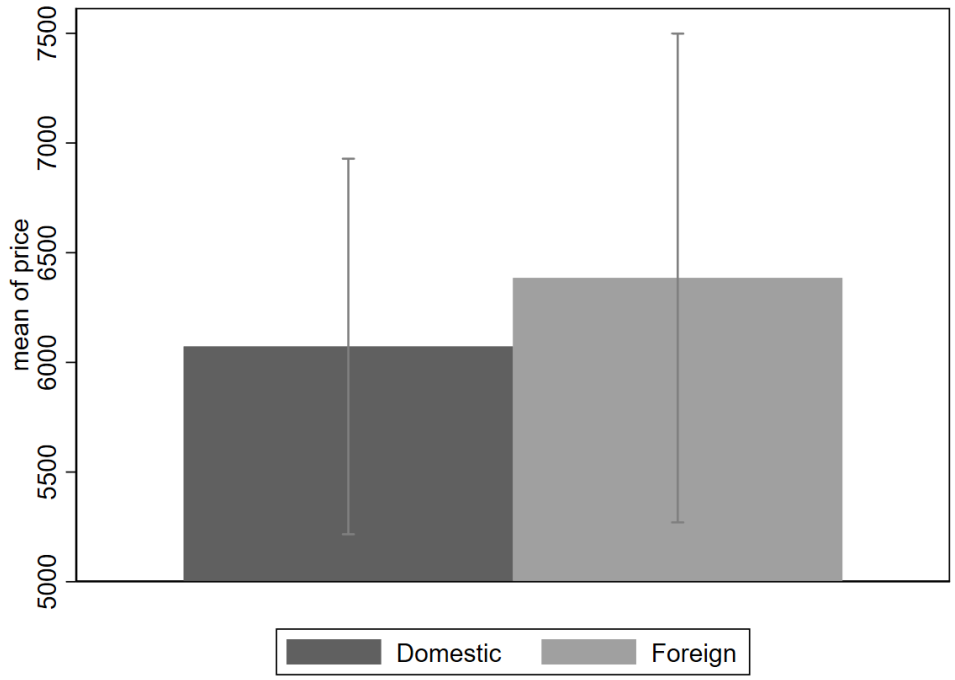
```
graph bar price, over(foreign)
```



CI Bar

- *cibar* (confidence interval bar) ist ein Ado und muss installiert werden

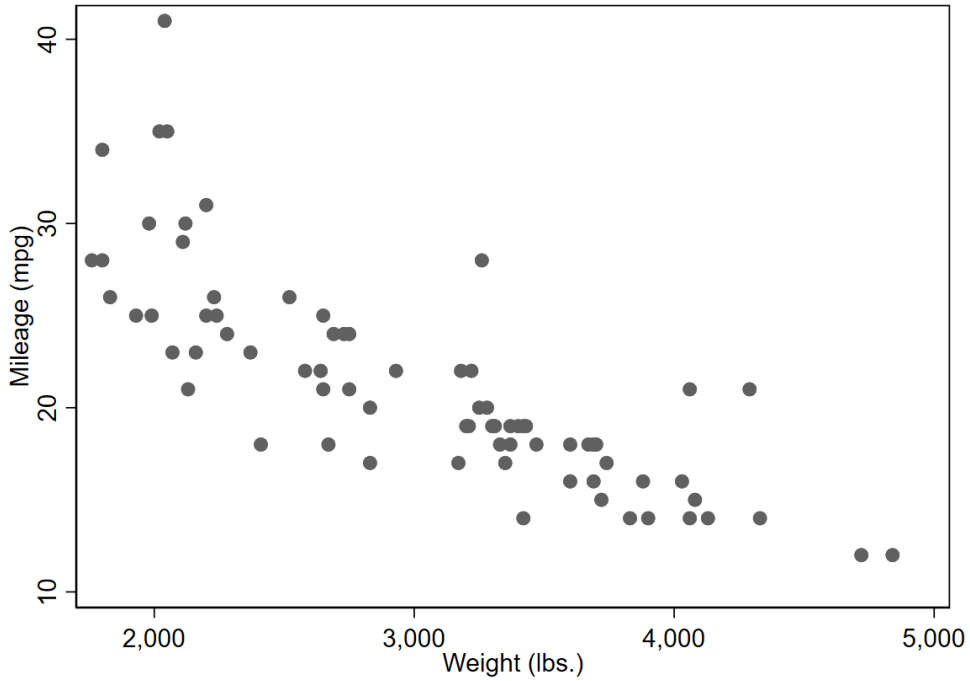
```
cibar price, over(foreign)
```



Assoziationen

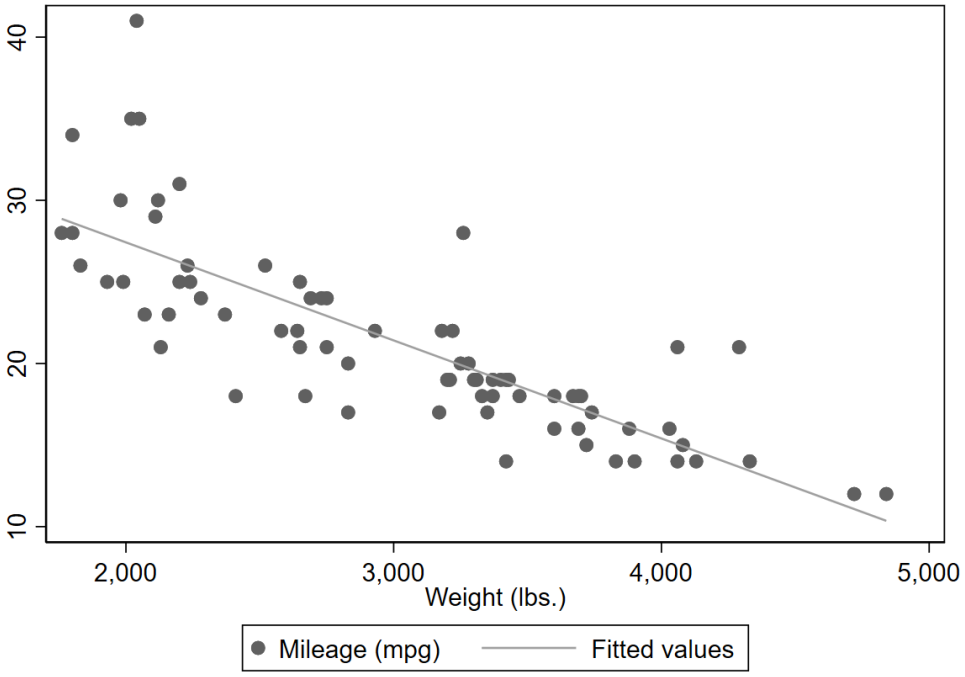
Scatter plot

```
scatter mpg weight
```



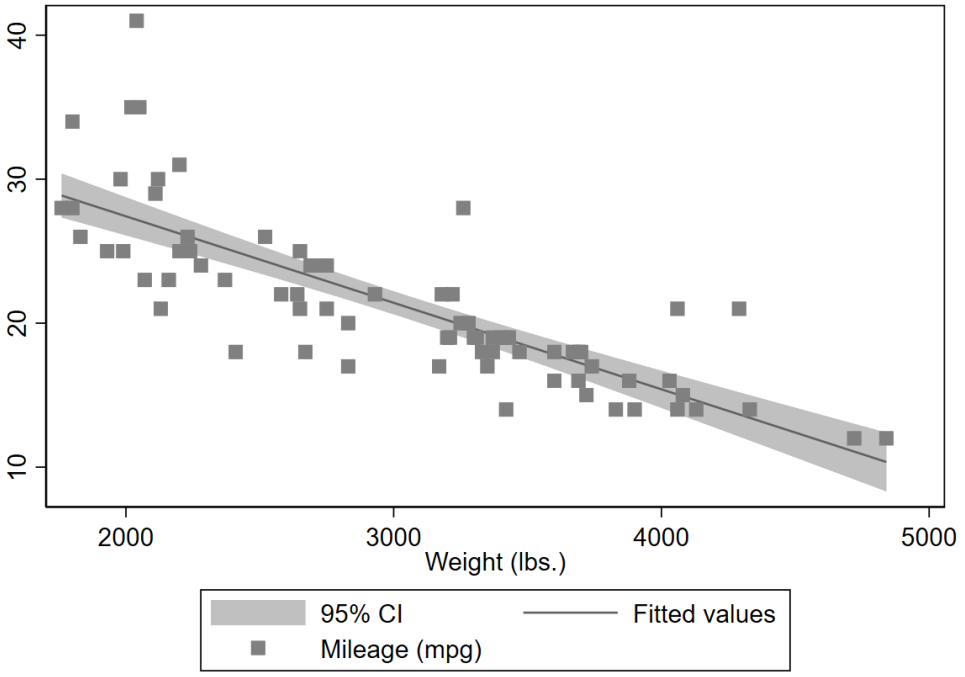
- mit Vorhersage eines linearen Zusammenhangs

```
twoway scatter mpg weight || lfit mpg weight
```



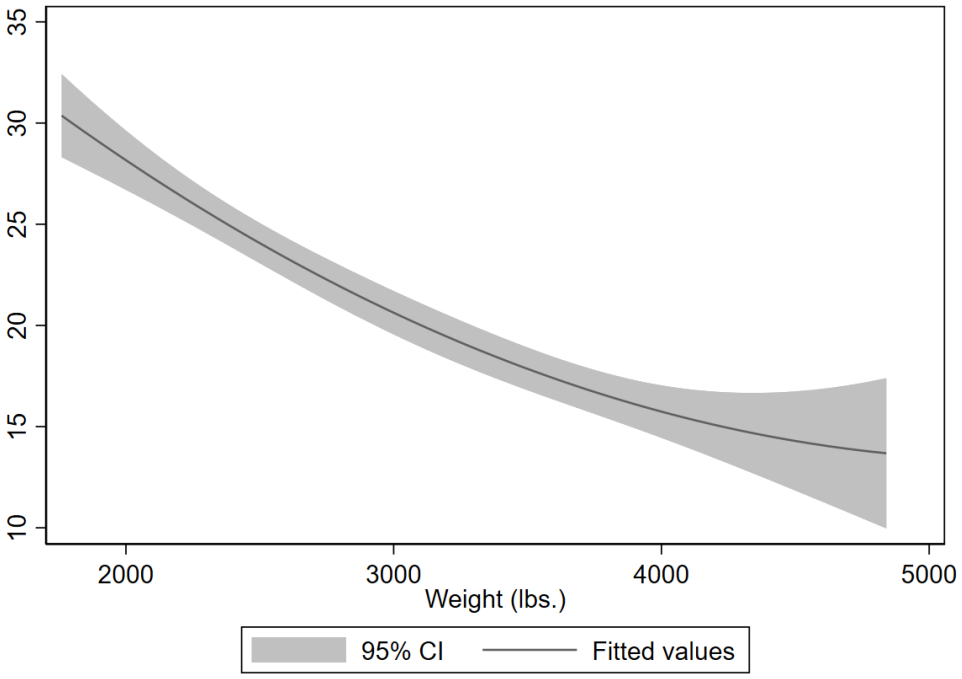
- mit CIs (confidence intervals)

```
twoway lfitci mpg weight || scatter mpg weight
```



- mit Vorhersage eines quadratischen Terms

```
twoway qfitci mpg weight
```



Many more options

- Diese Seite zeigt nur einige wichtigste Graphen, um u.a. Verteilungen zu visualisieren, noch mehr Infos zum erstellen von Graphen findet ihr auch im:
  - Stata Cheat Sheet
  - Stata Homepage: Visual overview for creating graphs

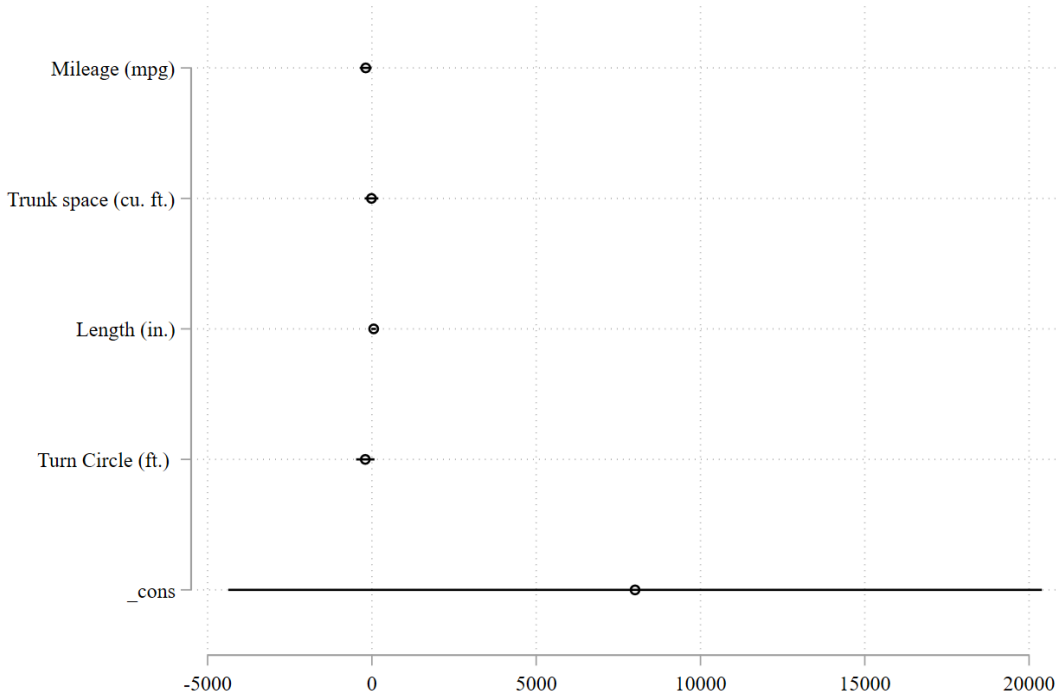
Coefplot

In Stata wird zumeist *Coefplot*, ein Ado von Ben Jann genutzt, um die Punktschätzer der Regression zu visualisieren, also die Koeffizienten zu plotten. Du musst also erst das Ado installieren, wenn du noch nie Coefplot genutzt hast:

```
ssc install coefplot, replace
```

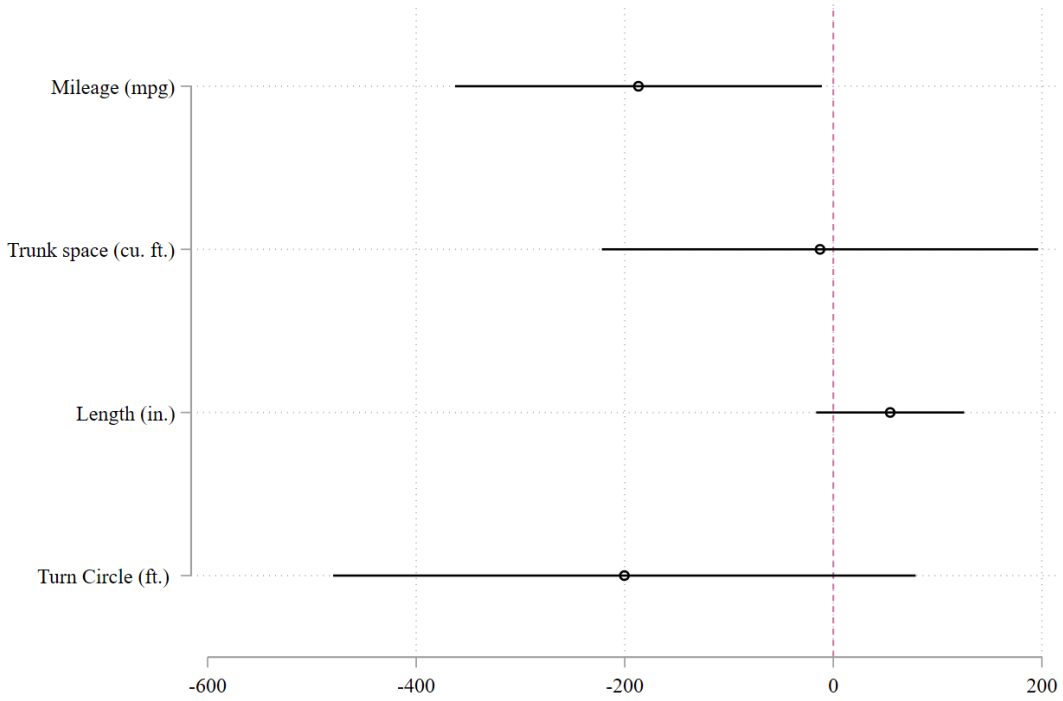
Coefplot ist ziemlich easy aufgebaut, nutze einfach den `coefplot` Befehl und die Punktschätzer der letzten berechneten Regression (oder eines anderen multivariaten Verfahrens) zu visualisieren.

```
sysuse auto, clear
quietly regress price mpg trunk length turn
coefplot
```



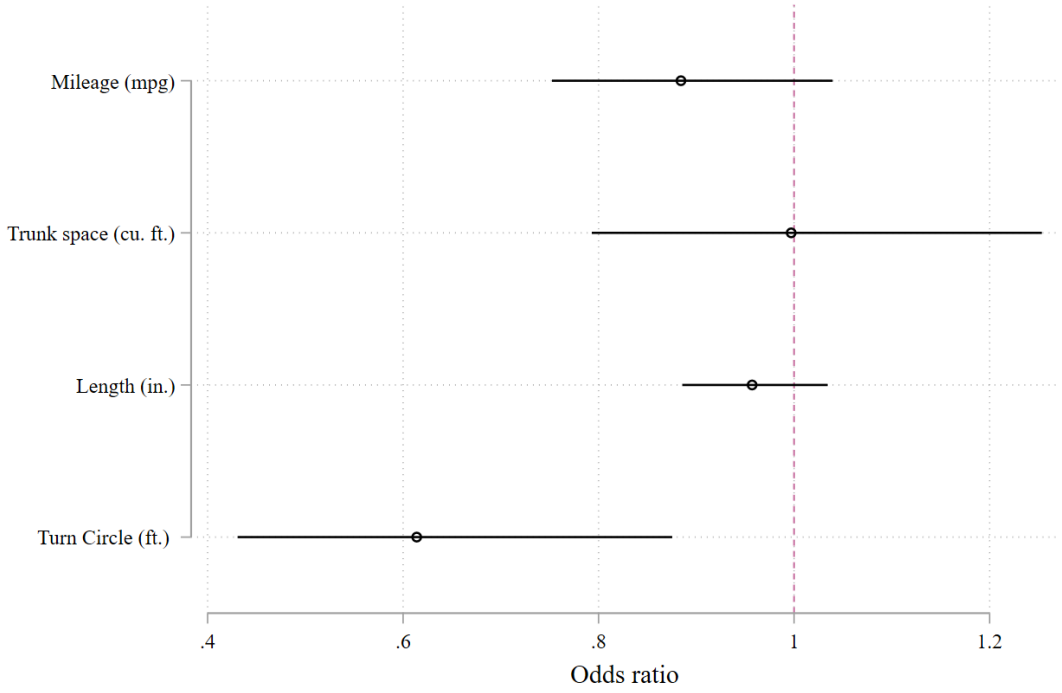
- Wie die Abbildung zeigt, macht es häufig keinen Sinn, die Konstante zu visualieren, da es sonst so aussieht, als ob die Punktschätzer der einzelnen Variablen gar keinen Effekt haben (und häufig sind wir ja nicht an der Konstante interessiert)
- Mit `drop(_cons)` wird die Konstante (oder auch andere Variablen wenn du die Variablenname hinzufügst) aus dem Coefplot ausgeschlossen
- `xline(0)` definiert wo die Markierungslinie sein soll. Hier sind wir daran interessiert, ob das Konfidenzinterval des Punktschätzers die Null beinhaltet, so erkennen wir auf den ersten Blick, ob ein Koeffizient signifikant ist

```
coefplot, drop(_cons) xline(0)
```



- Sollen andere Koeffizient visualisiert werden, kann es sein, dass Anpassungen an den Standardbefehl vorgenommen werden müssen
- Beim Logit können wir mit `eform` die Odds Ratios anstelle der Logit Koeffizient visualisieren
- Zur leichteren Interpretation setzen wir die Markierungslinie mit `xline(1)` auf 1

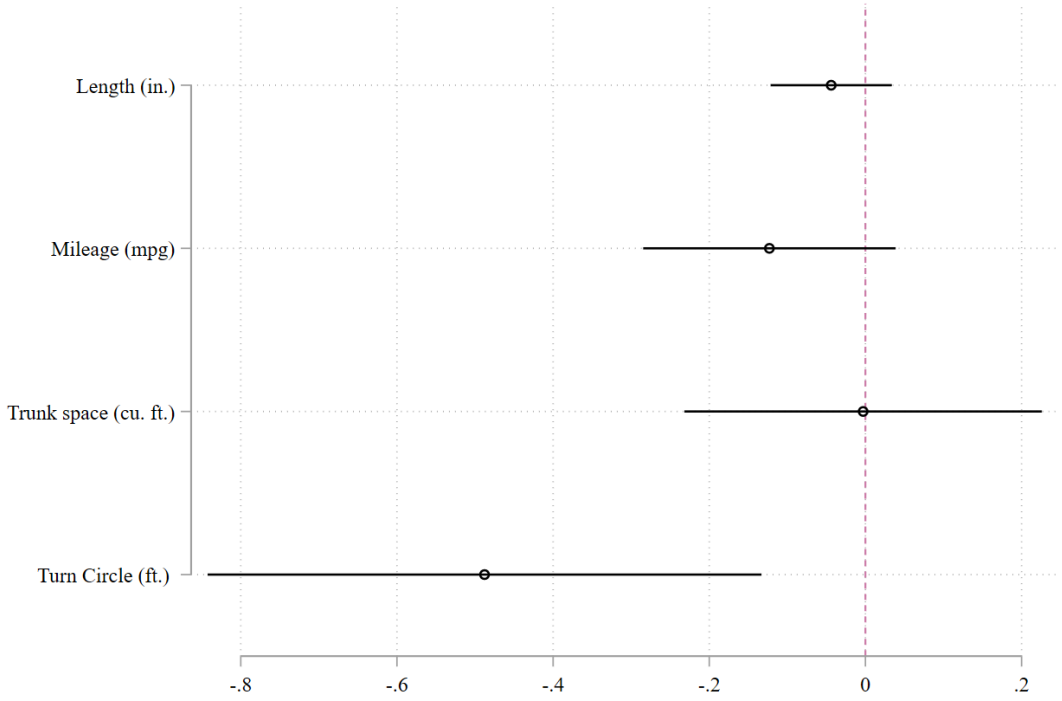
```
quietly logit foreign mpg trunk length turn
coefplot, drop(_cons) xline(1) eform xtitle(Odds ratio)
```



Sort and order

Zur leichteren Interpretation kann es auch hilfreich sein, Variablen neu zu ordnen, beispielsweise alphabetisch:

```
coefplot, drop(_cons) xline(0) ///
    order(length mpg t*)
```

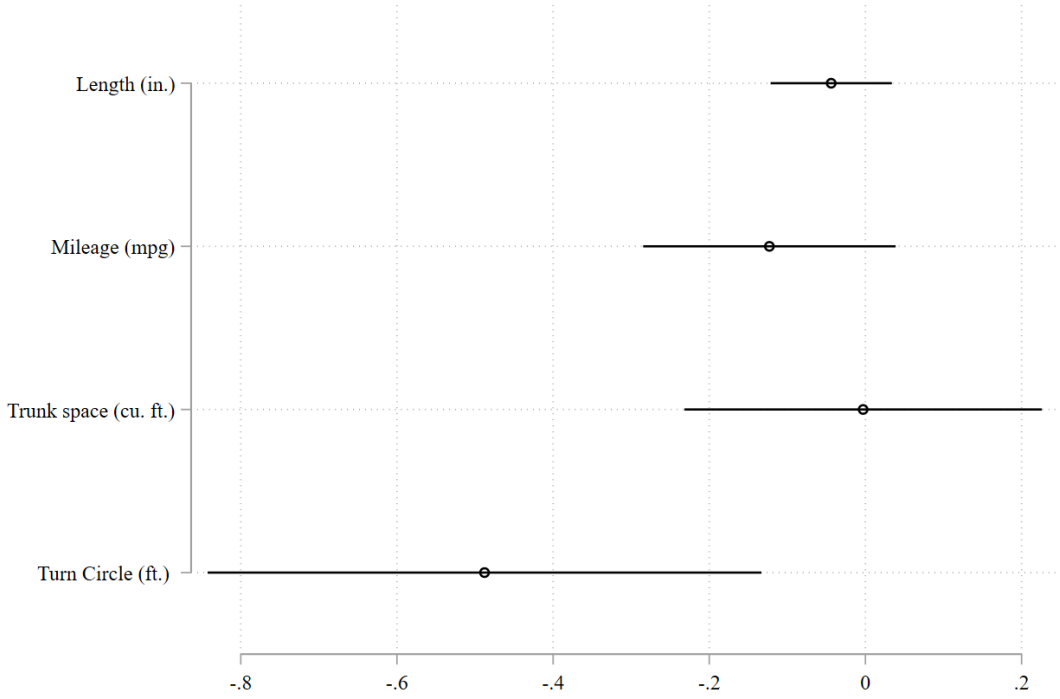


Nach der Größe der Effekte, in aufsteigender oder absteigender (`descending`) Reihenfolge:

```
*Absteigend mit
*coefplot, sort(, descending) drop(_cons)
*Aufsteigend:
coefplot, sort drop(_cons)
```

Oder auch nach dem Standardfehler der Koeffizienten:

```
coefplot, sort(, by(se)) drop(_cons)
```

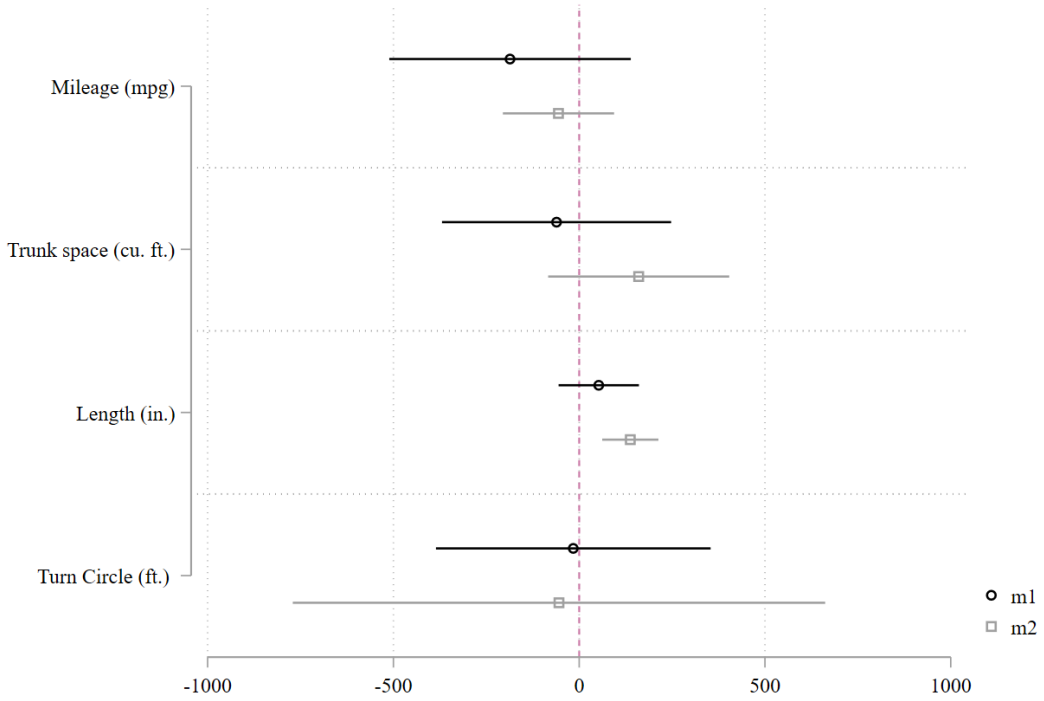


Modellvergleiche

Graphische Verfahren sind besonder gut geeignet, um Modelle miteinander zu vergleichen. Hierfür werden die Modelle zuerst gespeichert. Beispielsweise *estimates store m1* speichert das erste Model unter dem Namen `m1`.

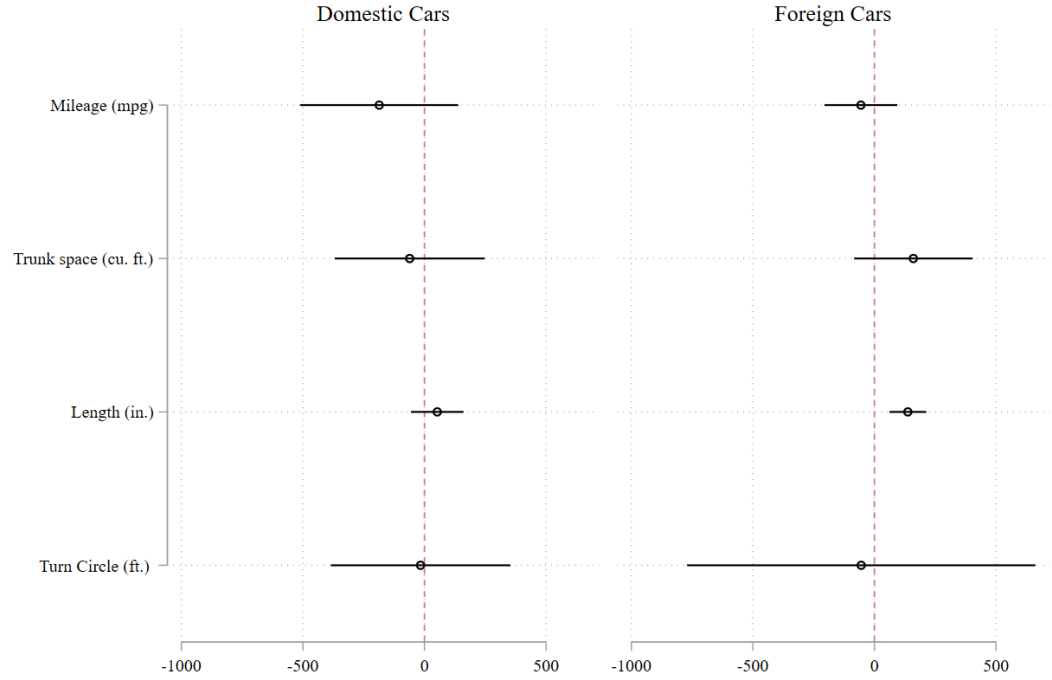
Alle gespeicherten Modelle werden sodann dem Coefplot Befehl hinzugefügt:

```
quietly regress price mpg trunk length turn if foreign==0
estimates store m1
quietly regress price mpg trunk length turn if foreign==1
estimates store m2

coefplot m1 m2, drop(_cons) xline(0)
```



Oder wir erstellen Subplots für unsere Modelle durch die Listung mit `||`. Mit dem Befehl `bylabel(Name)` geben wir den Modellen noch einen einschlagigen Namen.

```
coefplot m1, bylabel(Domestic Cars) ///
    || m2, bylabel(Foreign Cars)  ///
    ||, drop(_cons) xline(0)
```

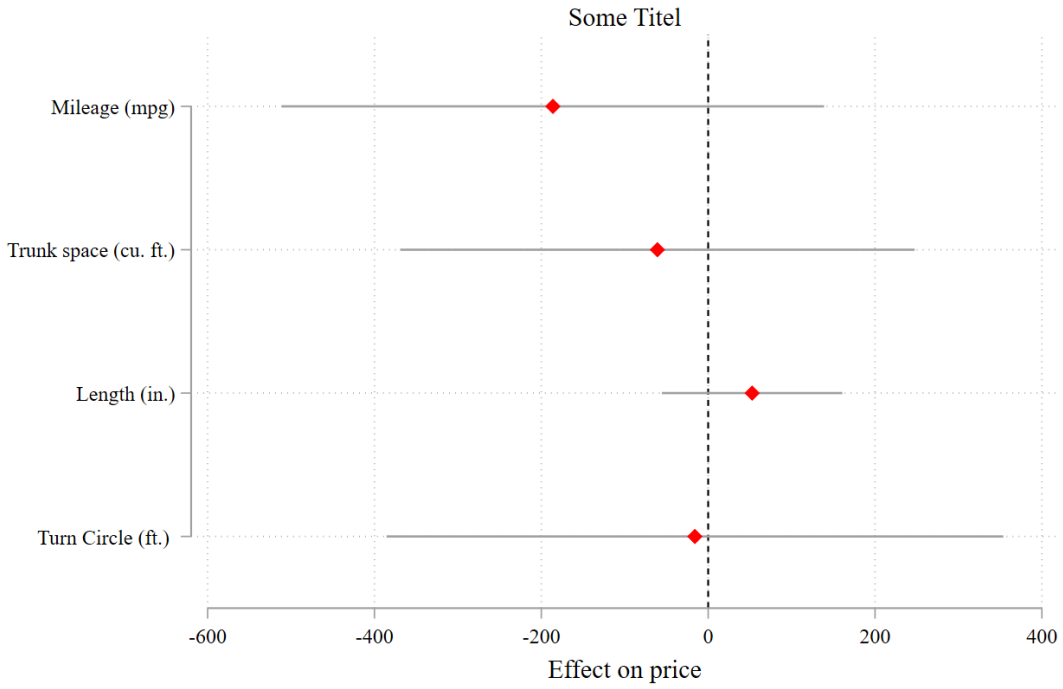


Further ado

Für eine Präsentation oder Abschlussarbeit lassen sich natürlich noch weitere graphische Aspekte anpassen:

- `lcolor`: Farbe der Markierungslinie
- `mcolor` und `msymbol` für die Farbe und das Symbol der Marker
- Labels der X Achse (`xtitle`) oder des Titel der Grafik (`title`)

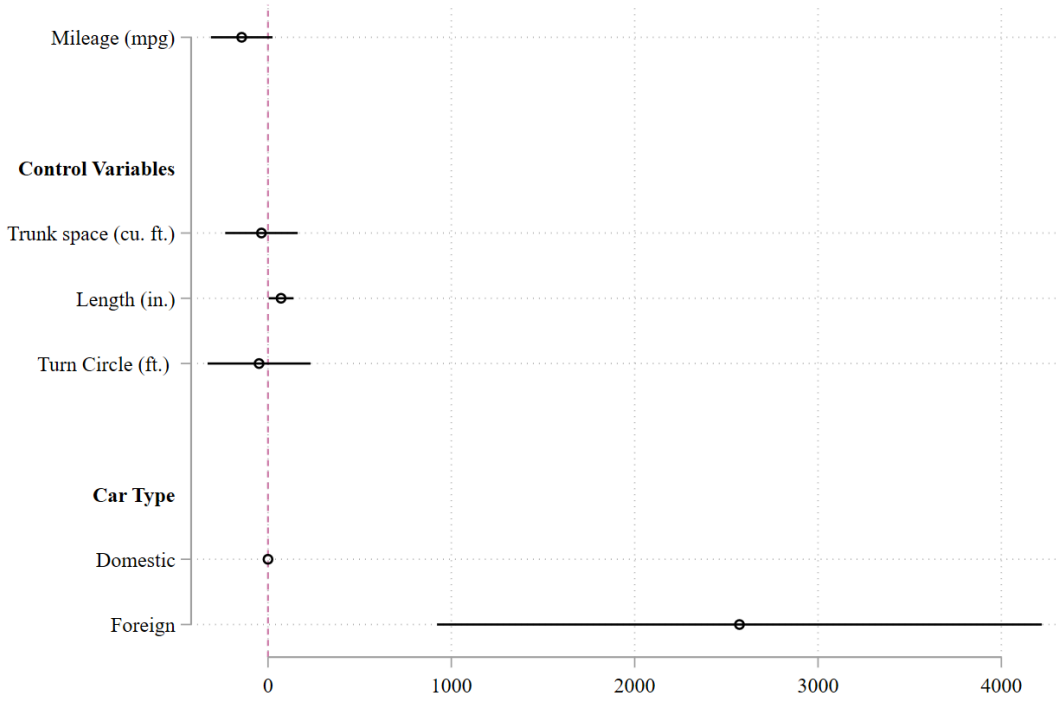
```
coefplot (m1, pstyle(p2)), drop(_cons) ///
    xline(0, lcolor(black)) ///
    mcolor(red) msymbol(D) ///
    xtitle(Effect on price) ///
    title("Some Titel")
```



Ebenso können `headings` eingebaut werden, beispielweise um Haupteffekte von Interaktioneffekten graphisch zu trennen. Oder, wie im nächsten Beispiel, um einen Haupteffekt von einer Reihe an Kontrollvariablen graphisch zu trennen.

Die Headings könnnen dabei direkt an der Stelle der Variablen oder an der Stelle der jeweiligen Ausprägung einer Variable (wie bei `0.foreign`) ausgegeben werden. Mit `omitted baselevels` wird die Ausgabe der Referenzkategorie unterdrückt.

```
quietly regress price mpg trunk length turn i.foreign

coefplot, xline(0) omitted baselevels drop(_cons) ///
    headings(trunk = "{bf:Control Variables}" ///
    0.foreign = "{bf:Car Type}")
```



Nicht das passende gefunden? Wahrscheinlich wirst du auf der Homepage von Ben Jann fündig, dort findet sich eine Vielzahl an Anpassungsmöglichkeiten, hier waren nur einige der wichtigsten Punkte aufgeführt.