Einführung in die Datenanalyse mit $\mathcal{R}$
Weiterführende Hinweise

Dag Tanneberg*

25. Oktober 2018

Inhaltsverzeichnis

1 Quellcode der Grafiken aus Sitzung 4 **1**
1.1 Small multiples 2
1.2 Selektive Ansprache von Aesthetics 3

2 Auswertung multiplikativer Interaktionsterme **6**
2.1 Auswertung von Vorhersagewerten 7
2.2 Marginal Effect Plots 9

1 Quellcode der Grafiken aus Sitzung 4

Sitzung 4 lieferte einen knappen Aufriss der Visualisierung statistischer Daten mit dem Paket ggplot2. Auf den hinteren Folien befinden sich zwei Abbildungen, nach deren Vorbild bestimmte Darstellungsmöglichkeiten recherchiert und erprobt werden sollten. Der folgende Abschnitt zeigt mit Hilfe kommentierten Quellcodes, wie man die beiden Grafiken erstellt. Bevor der Code erfolgreich ausgeführt werden kann, müssen die Pakete ggplot2 und gapminder geladen werden.

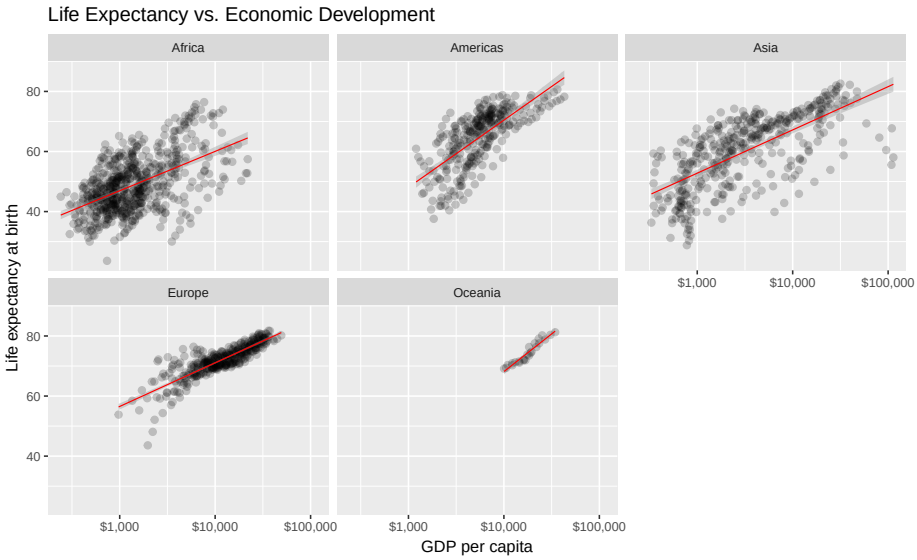
```
packs <- c("ggplot2", "gapminder")
# Does your system lack any of the packages stated in packs?
if (any(!(packs %in% installed.packages()))) {
  mis_pack <- setdiff(packs, installed.packages())
  cat("Missing package detected. Please install:", mis_pack)
  rm(mis_pack)
} else {
  for (i in packs) {
    library(i, character.only = TRUE)
```

*dag.tanneberg@uni-potsdam.de

```
  }
}
rm(i, packs)
data(gapminder)
```

1.1 Small multiples

Die erste Grafik zeigt mehrere Streudiagramme. Sie tragen für jeden der fünf Kontinente die durchschnittliche Lebenserwartung gegen den ökonomischen Entwicklungsstand ab. Darüber hinaus zeigt jedes Diagramm die lineare Korrelation beider Variablen und die Sichtbarkeit aller Datenpunkte wurde reduziert.



Das Besondere der Grafik ist die Verwendung sogenannter *small multiples*, d. h. kleinerer Darstellungen desselben Zusammenhangs, die auf die Ausprägungen mindestens einer weiteren Variable bedingen.[1] Wer mit Stata vertraut ist, hat in der Vergangenheit vielleicht ähnliche Grafiken mit der Option by() erstellt. In der Terminologie von ggplot2 handelt es sich um *facet*s und sie dienen der Organisation einer Reihe von Grafiken. Es gilt, zwei Anwendungsfälle zu unterscheiden:

1. Ziel ist eine eindimensionale Sequenz von Teilabbildungen. Hierfür kommt die Funktion facet_wrap() zum Einsatz.

2. Ziel ist eine Matrix statistischer Grafiken, deren Reihen und Spalten die Ausprägungen diskreter Variablen abtragen. Hierfür kommt die Funktion facet_grid() zum Einsatz.

[1] Wie vieles andere in der Visualisierung statistischer Daten geht der Begriff auf Edward Tufte zurück: Tufte 2013, 67.

In beiden Fällen werden Teilgrafiken durch die Verwendung eines Formelarguments definiert. Für facet_wrap verwendet man eine einseitige Formel, d. h. die linke Seite der Formel bleibt leer und rechts stehen eine oder mehrere durch '+' voneinander getrennte, kategoriale Variablen. Die Funktion facet_grid verwendet hingegen eine zweiseitige Formel, die zuerst Reihen und dann Spalten festlegt. Aber Beispiele sagen mehr als tausend Worte:

```
facet_wrap(~ continent)
facet_wrap(~ country + continent)
facet_grid(year ~ country)
```

Der nachstehende Block zeigt den vollständigen Quellcode der Grafik.

```
ggplot(
  data = gapminder,
  mapping = aes(x = gdpPercap, y = lifeExp)
) +
  geom_point(
    # This is simple example for how you set an aesthetic, rather
    # than mapping it to a variable. In this case we set
    # visibility of ALL data points to 20 per cent.
    alpha = 0.2
  ) +
  geom_smooth(
    # geom_smooth adds smoothers to your scatter plot. There is a
    # number of different smoothers available. This example uses
    # a linear regression model of the form y ~ x. For complete
    # information see:
    # https://ggplot2.tidyverse.org/reference/geom_smooth.html.
    method = 'lm', colour = "red", size = 0.3
  ) +
  scale_x_log10(labels = scales::dollar) +
  labs(
    x = "GDP per capita",
    y = "Life expectancy at birth",
    title ="Life Expectancy vs. Economic Development"
  ) +
  facet_wrap(
    # This part is tricky. I provide further information in the
    # accompanying text. For now, let it suffice to say that I
    # order ggplot2 to
    ~ continent, # draw a separate plot for each continent, and
    nrow = 2 # arrange them in 2 rows.
  ) +
  theme_gray(base_size = 9)
```

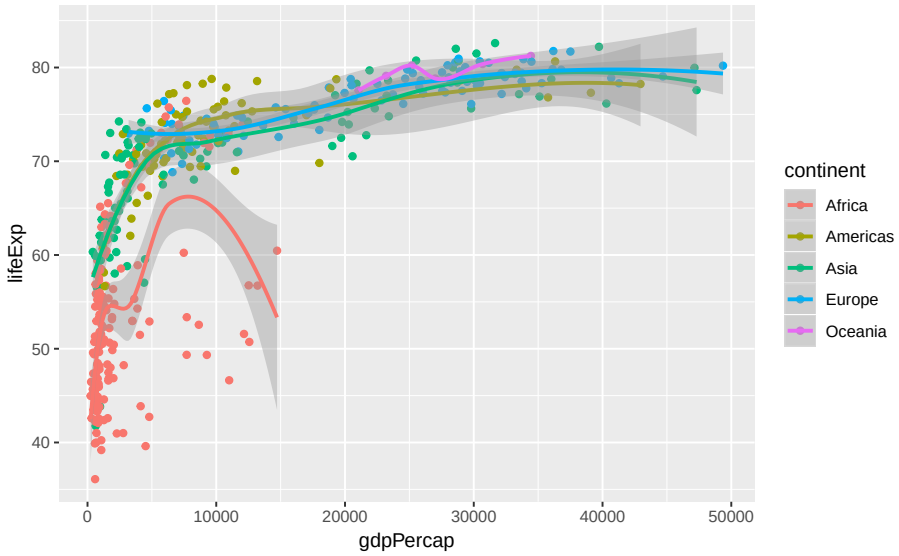
1.2 Selektive Ansprache von Aesthetics

Die zweite Abbildung zeigt erneut ein Streudiagramm von Lebenserwartung und ökonomischem Entwicklungsstand samt Glättung. Neu hinzu kommen eine Beschränkung auf den Zeitraum ab 1997 sowie eine Färbung nach Kontinent. Im Vergleich zum vorhergehenden Abschnitt wirkt diese Abbildung simpel. Der Teufel steckt im Detail, denn die durch die Werte der Variablen continent definierten Farbgruppen sollen zwar im Streudiagramm, nicht jedoch in der Glättung auftauchen.

Der folgende Codeblock zeigt die Deklaration des Plot-Objekts. Darin wird dem Argument colour die Variable continent übergeben. Es handelt sich um einen normalen mapping-Vorgang. Man könnten denken, die Sache sei mit dem anschließenden Aufruf von geom_point() und geom_smooth() erledigt.

```
p <- ggplot(
  data = gapminder[gapminder$year >= 1997, ], # Filter obs.
  mapping = aes(x = gdpPercap, y = lifeExp, colour = continent)
)
p + geom_point() + geom_smooth()

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric
= parametric, : Chernobyl! trL>n 6
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric
= parametric, : Chernobyl! trL>n 6
## Warning in sqrt(sum.squares/one.delta): NaNs produced
## Warning in stats::qt(level/2 + 0.5, pred$df): NaNs produced
```



Der Schein trügt, denn das Argument colour greift für *alle* Schichten der Grafik. Folglich zeichnet geom_smooth() für jeden Kontinent eine eigene Glättung ein.[2] Die Lösung des Problems liegt in der selektiven Zuordnung einer *Aesthetic*. Das kann in diesem Fall auf zwei Arten geschehen.

1. Das Argument colour verschwindet aus dem Aufruf von ggplot() und wird erst beim Aufruf von geom_point() übergeben.

2. Das Argument colour verbleibt im Aufruf von ggplot() und wird aus dem Aufruf von geom_smooth gelöscht.

```
# Variante 1: Selektive Übergabe eines Aesthetics
p <- ggplot(
  data = gapminder[gapminder$year >= 1997, ], # Filter obs.
  mapping = aes(x = gdpPercap, y = lifeExp)
) +
  geom_point(mapping = aes(colour = continent)) +
  geom_smooth()
# Variante 2: Löschen eines Aesthetics
p <- ggplot(
  data = gapminder[gapminder$year >= 1997, ], # Filter obs.
  mapping = aes(x = gdpPercap, y = lifeExp, colour = continent)
) +
  geom_point() +
  geom_smooth(mapping = aes(colour = NULL))
```

[2] Die zahlreichen Warnmeldungen haben sämtlich damit zu tun, dass die Fallzahlen einiger Kontinente sehr klein sind.

Der vorstehende Block demonstriert beide Möglichkeiten. Welcher man den Vorzug gibt, hängt vom konkreten Anwendungsfall ab. Sollen beispielsweise weitere Geometrien das Argument colour verwenden, dann streicht man es am besten nur aus dem Smoother. Interessiert colour hingegen nur in einer einzigen Geometrie, dann sollte es aus dem Aufruf von ggplot() verschwinden und anschließend einzeln übergeben. Das nachfolgende, abschließende Listing zeigt den vollständigen Quellcode der Grafik.

```
ggplot(
  data = gapminder[gapminder$year >= 1997, ],
  mapping = aes(x = gdpPercap, y = lifeExp, colour = continent)
) +
  geom_point(alpha = 0.6) +
  geom_smooth(method = "loess", aes(colour = NULL)) +
  scale_x_log10(labels = scales::dollar)  +
  labs(
    x = "GDP per capita",
    y = "Life expectancy at birth",
    colour = "Continent",
    title = "Life Expactancy vs. Economic Development",
    subtitle = "Graph shows 1997, 2003, and 2007."
  ) +
  theme_gray(base_size = 9)
```

2 Auswertung multiplikativer Interaktionsterme

Die letzte Sitzung der Veranstaltung riss die Schätzung und Auswertung generalisierter linearer Modelle (GLM) an. Sie griff zu diesem Zweck auf einen Datensatz zurück, der Informationen über alle namentlich erwähnten Charaktere der Fantasyreihe "A Song of Ice and Fire" bereitstellte. Die Fragestellung lautete, mit welcher Wahrscheinlichkeit der Hauptcharakter Jon Snow sterben würde. Zu ihrer Beantwortung griff die Veranstaltung auf Angaben über die Zugehörigkeit zu einem Adelshaus, das Alter[3], das Geschlecht sowie den Adelsstatus zurück. Das logistische Regressionsmodell verwendete außerdem eine Interaktion von Geschlecht und Adelsstatus sowie ein kubisches Polynom des Alters. Der nachstehende Block wiederholt die Modellspezifikation.

```
fit <- glm( # generate model
  died ~ 0 + allegiances +
    gender * nobility +
```

[3] In Ermangelung besserer Informationen misst die entsprechende Variable nicht das natürliche Alter, sondern zählt die Anzahl der seit Einführung des Charakters verstrichenen Kapitel bis zu seinem Tod. Mit etwas gutem Willen handelt es sich um eine Messung der Prozesszeit.

```
    age_in_chapters + I(age_in_chapters^2) + I(age_in_chapters^3),
  family = binomial(link = "logit"),
  data = asoiaf
)
```

Offen blieb zum Schluss die Auswertung multiplikativer Interaktionsterme. Puristen verwenden zu diesem Zweck partielle Ableitungen und Kommunikatoren setzen gerne auf den Vergleich von Vorhersagewerten. Aber die meisten Sozialwissenschaftler schwören auf *Marginal Effect Plots* (MEPs). Partielle Ableitungen sind im Falle generalisierter linearer Modelle nicht besonders aussagekräftig. Da man dafür außerdem $\mathcal{R}$ nicht benötigt, verzichte ich auf ihre Darstellung und verweise auf den Klassiker Kam und Franzese 2007. Das folgende Kapitel bespricht zunächst den Umgang mit Vorhersagewerten und abschließend MEPs.[4]

2.1 Auswertung von Vorhersagewerten

Im Wesentlichen gleicht die Auswertung von Vorhersagewerten den Verfahren, die bereits im Rahmen der letzten Sitzung vorgestellt wurden. Sie erfolgt in drei Schritten.

1. Zuerst muss das entsprechende Modell geschätzt werden.
2. Der Anwender definiert dann die vorherzusagenden Szenarien und speichert sie in einem eigenen Data Frame.
3. Mit Hilfe der Funktion predict() und ihres Arguments newdata generiert man die Vorhersagewerte und fasst sie abschließend durch Tabellen oder Abbildungen zusammen.

Die einzige Komplikation erwächst im zweiten Schritt, da die vorherzusagenden Szenarien möglichst den gesamten Eigenschaftsraum der Interaktion abbilden sollten. Dabei hilft die Funktion expand.grid(). Sie erwartet Vektoren als Argumente und generiert einen Data Frame, in dem jede Reihe einer Kombination der übergebenen Vektoren entspricht. Der resultierende Datensatz hat so viele Reihen, wie es Kombinationen der Vektoren gibt. Der nachfolgende Block zeigt ein einfaches Beispiel.

```
expand.grid(x = 0:1, y = 0:1) # There are 2^2 combinations.

##   x y
## 1 0 0
## 2 1 0
## 3 0 1
## 4 1 1
```

[4] Die hier vorgestellten Auswertungsverfahren vermitteln erneut unbequeme, aber grundlegende Programmiertechniken. Wer es etwas bequemer mag, wirft bitte einen Blick auf das Paket effects von Jon Fox oder das Paket sjPlot von Daniel Lüdecke und Carsten Schwemmer.

Da in dem oben gezeigte Regressionsmodell Geschlecht und Adelsstatus miteinander interagieren, sollten mindestens deren Kombinationsmöglichkeiten voll ausgeschöpft werden. Zusätzlich zeigte sich im Rahmen der Sitzung eine starke implizite Interaktion mit der assoziierten Adelsfamilie. Daher lohnt es sich, auch diese umfassend zu berücksichtigen. Das Alter wird im Folgenden auf dem Median konstant gehalten. Der resultierende Datensatz enthält eine Zeile für jede der zwölf möglichen Kombinationen von **gender**, **nobility** und **allegiances**. Die Wert von **age_in_chapters** bleibt über alle Reihen hinweg konstant.

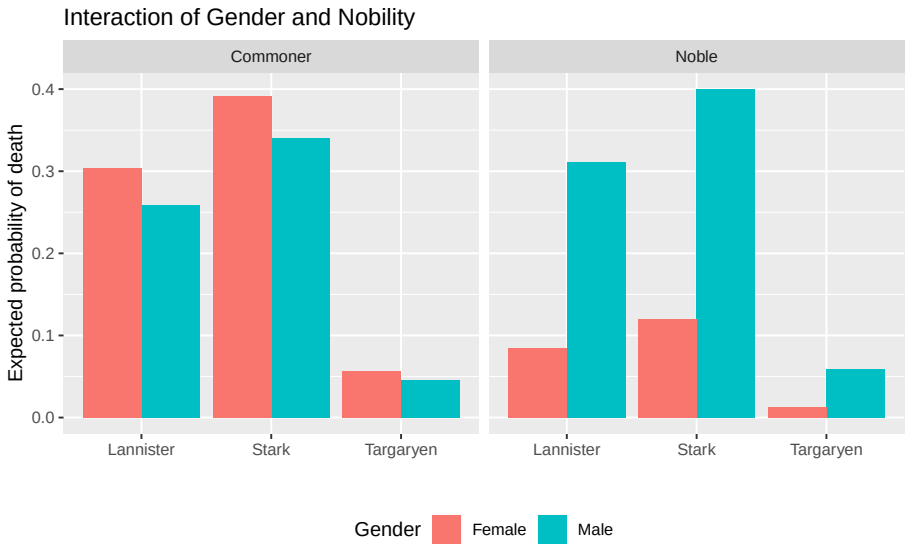
```
# Define scenarios
pred_data <- expand.grid(
  allegiances = c("Lannister", "Stark", "Targaryen"),
  nobility = 0:1, gender = 0:1,
  age_in_chapters = median(
    asoiaf[, "age_in_chapters"], na.rm = TRUE
  ), stringsAsFactors = FALSE
)
# Make predictions
pred_data[, "fitted"] <- predict(
  fit, newdata = pred_data, type = "response"
)
pred_data # Show data
```

```
##    allegiances nobility gender age_in_chapters     fitted
## 1    Lannister        0      0             126 0.30433574
## 2        Stark        0      0             126 0.39193082
## 3    Targaryen        0      0             126 0.05682782
## 4    Lannister        1      0             126 0.08476429
## 5        Stark        1      0             126 0.12006921
## 6    Targaryen        1      0             126 0.01259482
## 7    Lannister        0      1             126 0.25924574
## 8        Stark        0      1             126 0.34020963
## 9    Targaryen        0      1             126 0.04598431
## 10   Lannister        1      1             126 0.31161734
## 11       Stark        1      1             126 0.40010286
## 12   Targaryen        1      1             126 0.05868707
```

Zuletzt übergibt man die Vorhersageszenarian an die Funktion predict() und speichert sie in einem neuen Vektor.[5] Nach ein bißchen Feinschliff können die Ergebnisse in einem Säulendiagramm dargestellt werden. Alles spricht dafür, dass adlige Frauen bei George R. R. Martin die besten Überlebensaussichten genießen. Adlige Männer sterben dagegen auffällig häufiger.

[5] In der Regel generiert predict() mit dem Argument se.fit = TRUE sogar den Standardfehler des Vorhersagewerts.

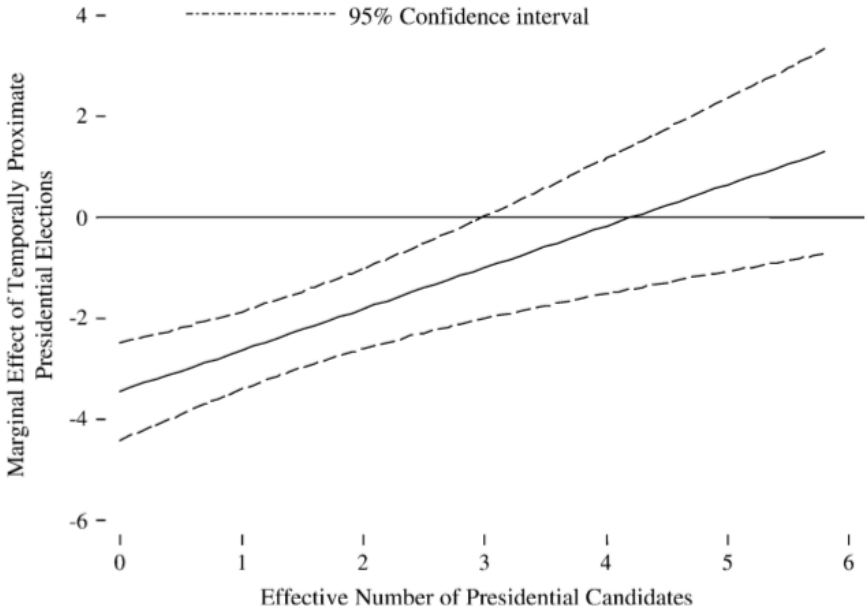
```r
pred_data <- within(pred_data, {
  # Note: within() requires curly brackets whenever you want to
  # evaluate multiple statements.
  gender_label <- factor(gender, 0:1, c("Female", "Male"))
  nobility_label <- factor(nobility, 0:1, c("Commoner", "Noble"))
  }
)
ggplot(
  data = pred_data,
  mapping = aes(x = allegiances, y = fitted, fill = gender_label)
) +
  geom_bar(stat = 'identity', position = "dodge") +
  facet_grid( ~ nobility_label) +
  labs(
    title = "Interaction of Gender and Nobility",
    x = "",
    y = "Expected probability of death",
    fill = "Gender"
  ) +
  theme(legend.position = "bottom")
```



2.2 Marginal Effect Plots

Häufig verwenden Sozialwissenschaftler sogenannte Marginal Effect Plots. Sie tragen den Wert des Regressionsgewichts eines substantiell interessierenden Prädiktors gegen den Wertebereich eines Moderators ab. Weiterhin zeichnen diese Darstellungen häufig Konfidenzintervalle um den substantiell interessierenden

Abbildung 1: Beispiel eines Marginal Effect Plots



Effekt herum ein. Daher erlaubt diese Art der Darstellung einerseits Aussagen über den Verlauf einer Interaktion. Andererseits vermittelt sie einen (kritikwürdigen[6]) Eindruck von der statistischen Signifikanz sowohl des Interaktionsterms als auch des dargestellten Effekts. Abbildung 1 reproduziert das Beispiel aus dem einflussreichen Beitrag Brambor, Clark und Golder (2005). Die Anfertigung solcher Grafiken ist nicht trivial, da die Berechnung des korrekten Standardfehlers erhebliches Kopfzerbrechen bereiten kann.

[6] Siehe Pepinsky 2018 mit weiteren Nachweisen.

In der einfachsten aller Welten, d. h. einer linearen Regression mit zwei interagierenden Variablen, liegen die Dinge vergleichsweise einfach. Es sei $y = \beta_0 + \beta_1 x + \beta_2 z + \beta_3 xz + \epsilon$ das interessierende Populationsmodell. Dann beschreibt $\frac{\partial y}{\partial x} = \beta_1 + \beta_3 z$ dessen partielle Ableitung nach x. Sie entspricht der aufsteigenden, durchgezogenen Linie in Abbildung 1. Dieser Teil ist im vorliegenden Beispiel schnell erledigt. Zunächst extrahiert man die notwendigen Koeffizienten aus dem Regressionsmodell. Dann erzeugt man Matrix mit den interessierenden Vorhersageszenarien und zuletzt wird ausmultipliziert.

```
mu <- coef(fit)[c("gender", "gender:nobility")]
X <- cbind(1, 0:1)
# Wir setzen Gender auf 1, denn der marginale Effekt ist
# beta_[gender] + beta_[nobility] * nobility
mu <- X %*% mu
# Hier werden Matrizen multipliziert. Insbesondere
# bei Interaktionstermen mit kontinuierlichen Variablen spart das
# Zeit. In unserem sehr einfachen Beispiel wäre auch Folgendes
# möglich:
coef(fit)["gender"] + c(0, 1) * coef(fit)["gender:nobility"]

## [1] -0.2231569  1.5867384
```

Der Standardfehler dieser Interaktion berechnet sich wie folgt:

$$\hat{\sigma}_{\frac{\partial y}{\partial x}} = \sqrt{var(\beta_1) + z^2 var(\beta_3) + 2z cov(\beta_1 \beta_3)}$$

Da nur zwei Variablen miteinander interagieren, fällt die Formel vergleichsweise übersichtlich aus. Die Berechnung in $\mathcal{R}$ kommt jedoch keinesfalls handzahm daher. Eine mögliche Schrittfolge sieht so aus:

1. Extrahiere die Varianz-Kovarianz-Matrix der Effektschätzer. Das geschieht mit der Funktion vcov().
2. Greife auf die Varianzterme der interessierenden Koeffizienten zu. Das geschieht mit Hilfe der Funktion diag() und anschließender (Namens-)Indizierung des resultierenden Vektors.
3. Extrahiere den Koviarianzterm durch Indizierung der Varianz-Kovarianz-Matrix.
4. Berechne den Standardfehler für die verschiedenen Szenarien.

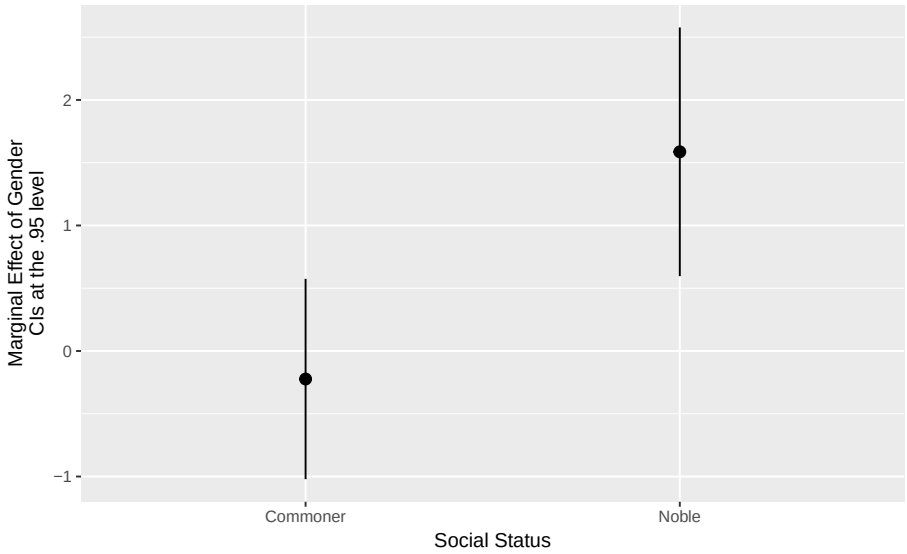
Der nachstehende Block zeigt die einzelnen Schritte in $\mathcal{R}$-Code.

```
# Step 1:
sigma2 <- vcov(fit)
# Step 2 & 3:
sigma2_coefs <- diag(sigma2)[c("gender", "gender:nobility")]
```

```r
# Step 4:
cov_coefs <- sigma2["gender", "gender:nobility"]
# Step 5:
se <- sigma2_coefs["gender"] +
  c(0, 1)^2 * sigma2_coefs["gender:nobility"] +
  2 * c(0, 1) * cov_coefs
se <- sqrt(se)
```

Abschließend fasst man die Ergebnisse in einem Data Frame zusammen, berechnet die Konfidenzintervalle und übergibt die Informationen an ggplot2. Erneut zeigt sich ein klarer Überlebensvorteil für adlige Frauen.

```r
maref_dta <- data.frame(mu = mu, se = se, nobility = 0:1)
maref_dta <- within(maref_dta, {
  nobility_label <- factor(
    nobility, 0:1, label = c("Commoner", "Noble")
  )
  lower <- mu - 1.96 * se
  upper <- mu + 1.96 * se
  }
)
ggplot(
  data = maref_dta,
  aes(x = nobility_label, y = mu, ymin = lower, ymax = upper)
) +
  geom_pointrange() +
  labs(
    x = "Social Status",
    y = "Marginal Effect of Gender\nCIs at the .95 level"
  )
```



Es folgt eine abschließende Mahnung an den Leser. Das hier vorgestellte Verfahren dient vor allem didaktischen Zwecken. Es ist hochgradig fehleranfällig und deshalb insbesondere für Einsteiger *nicht* geeignet. Die Verallgemeinerung auf komplexe Interaktionsterme oder die Verwendung von robusten Standardfehlern ist möglich, aber nicht trivial. Die Verwendung eines der oben genannten Pakete sei nachdrücklich empfohlen. Alternativ können auch Simulationsmethoden zum Einsatz kommen. King, Tomz und Wittenberg 2000 sowie das Standardwerk Gelman und Hill 2006 bieten einen guten Einblick.

Literatur

Brambor, Thomas, William Roberts Clark und Matt Golder. 2005. "Understanding Interaction Models: Improving Empirical Analyses". *Political Analysis* 14, Nr. 1 (): 6382. doi:10.1093/pan/mpi014.

Gelman, Andrew, und Jennifer Hill. 2006. *Applied Regression and Multilevel/Hierarchical Models*. Cambridge; New York: Cambridge University Press. ISBN: 978-0-521-68689-1.

Kam, Cindy D., und Robert J. Franzese. 2007. *Modeling and Interpreting Interactive Hypotheses in Regression Analysis*. Ann Arbor: University of Michigan Press. ISBN: 0-472-06969-1.

King, Gary, Michael Tomz und Jason Wittenberg. 2000. "Making the Most of Statistical Analyses: Improving Interpretation and Presentation". *American Journal of Political Science* 44, Nr. 2 (): 341–355. ISSN: 0092-5853.

Pepinsky, Thomas B. 2018. "Visual Heuristics for Marginal Effects Plots". *Research & Politics* 5, Nr. 1 (): 205316801875666. ISSN: 2053-1680, 2053-1680, besucht am 25. 10. 2018. doi:10.1177/2053168018756668. http://journals.sagepub.com/doi/10.1177/2053168018756668.

Tufte, Edward R. 2013. *Envisioning Information*. Fourteenth printing. OCLC: 892660693. Cheshire, Connecticut: Graphics Press. ISBN: 978-0-9613921-1-6.