



Institut für Empirie & Statistik  
der FOM Hochschule  
für Oekonomie & Management

# **A Student's Guide to R**

**Übersetzung des englischen Originals**

---

Bianca Krol, Sebastian Sauer, Roger Bons, Oliver Gansser, Matthias Gehrke, Herbert Hollmann, Tanja Kistler, Andreas Kladroba und Thomas Weiß

**Stand 2021-07-09**

# Inhaltsverzeichnis

<b>Vorwort</b>	<b>5</b>
<b>1 Einleitung</b>	<b>7</b>
<b>2 Los geht's mit RStudio</b>	<b>9</b>
2.1 Mit einem RStudio-Server verbinden . . . . .	11
2.2 Arbeiten mit Dateien . . . . .	16
2.3 Die Funktionen der Panel und Reiter . . . . .	19
<b>3 Eine metrische Variable</b>	<b>22</b>
3.1 Numerische Zusammenfassungen . . . . .	22
3.2 Grafische Zusammenfassungen . . . . .	24
3.3 Dichtekurven . . . . .	29
3.4 Häufigkeitspolygone . . . . .	30
3.5 Normalverteilungen . . . . .	30
3.6 Inferenz einer einzelnen Stichprobe . . . . .	31
<b>4 Eine kategoriale Variable</b>	<b>34</b>
4.1 Analyse kategorialer Daten . . . . .	34
4.2 Der Binomialtest . . . . .	35
4.3 Der Anteilswerttest . . . . .	37
4.4 Anpassungstests . . . . .	38
<b>5 Zwei metrische Variablen</b>	<b>42</b>
5.1 Scatterplots . . . . .	42
5.2 Korrelationen . . . . .	43
5.3 Paarweise Plots . . . . .	44
5.4 Einfache lineare Regression . . . . .	45
<b>6 Zwei kategoriale Variablen</b>	<b>52</b>
6.1 Kreuztabellen . . . . .	52
6.2 Tabellen erstellen . . . . .	55
6.3 Chi-Quadrat-Test . . . . .	55
6.4 Exakter Test nach Fisher . . . . .	57

<b>7</b>	<b>Metrische Antwortvariable, kategorialer Prädiktor</b>	<b>58</b>
7.1	Eine binäre Variable als Prädiktor: numerische und grafische Zusammenfassung . . . . .	58
7.2	Ein dichotomer Prädiktor: Zweistichproben-t-Test . . . . .	60
7.3	Nichtparametrische Zweistichprobentests . . . . .	61
7.4	Permutationstest . . . . .	62
7.5	Einfaktorielle Varianzanalyse . . . . .	64
7.6	Tukeys Post-hoc-Test . . . . .	66
<b>8</b>	<b>Kategoriale Antwortvariable, metrischer Prädiktor</b>	<b>68</b>
8.1	Logistische Regression . . . . .	68
<b>9</b>	<b>Überlebenszeitanalysen</b>	<b>71</b>
9.1	Kaplan-Meier-Kurve . . . . .	71
9.2	Cox-Modell mit proportionalem Ausfallrisiko . . . . .	72
<b>10</b>	<b>Mehr als zwei Variablen</b>	<b>73</b>
10.1	Zwei- (oder mehr-) faktorielle ANOVA . . . . .	73
10.2	Multiple Regression . . . . .	75
<b>11</b>	<b>Wahrscheinlichkeitsverteilungen und Zufallsvariablen</b>	<b>81</b>
<b>12</b>	<b>Power-Berechnungen</b>	<b>86</b>
12.1	Vorzeichen-Test . . . . .	86
12.2	t-Test . . . . .	88
<b>13</b>	<b>Datenmanagement</b>	<b>90</b>
13.1	Überprüfung von Dataframes . . . . .	90
13.2	Hinzufügen neuer Variablen zu einem Dataframe . . . . .	91
13.3	Variablen löschen . . . . .	93
13.4	Variablen umbenennen . . . . .	93
13.5	Erstellen von Teilmengen spezifischer Beobachtungen . . . . .	95
13.6	Sortieren von Dataframes . . . . .	96
13.7	Zusammenfügen von Dataframes . . . . .	97
13.8	Extrahieren und Zusammenfassen von Informationen . . . . .	98
13.9	Neue Variablen hinzufügen . . . . .	99
13.10	Gruppenstatistiken . . . . .	102
13.11	Umgang mit fehlenden Werten . . . . .	103
<b>14</b>	<b>Fallstudie Gesundheitsevaluation (HELP-Studie)</b>	<b>106</b>
<b>15</b>	<b>Aufgaben</b>	<b>108</b>
<b>16</b>	<b>Hinweise</b>	<b>112</b>

## Vorwort

In 2016 hat die FOM Hochschule für Oekonomie & Management sich auf den Weg gemacht, in der Methodenausbildung die Kompetenz der Data Literacy zu fokussieren. In den zurückliegenden Semestern wurde die Ausbildung daher in Anlehnung an die [GAISE](#)(Guidelines for Assessment and Instruction in Statistics Education)-Empfehlungen modernisiert. 2019 wurde die FOM aufgrund dieses Ausbildungskonzeptes in das [Data Literacy Education Netzwerk des Stifterverbandes](#) aufgenommen und zu Beginn dieses Jahres hat die FOM die [Data-Literacy-Charta](#) unterschrieben, in der Folgendes zu finden ist: “Data Literacy umfasst die Datenkompetenzen, die für alle Menschen in einer durch Digitalisierung geprägten Welt wichtig sind. Sie ist unverzichtbarer Bestandteil der Allgemeinbildung.”

Diesem Anspruch versuchen wir im Rahmen unserer Methodenausbildung gerecht zu werden. Um daten-literat zu werden, liegt der Fokus unserer Ausbildung auf dem konzeptionellen Verstehen, der Betonung des gesamten Analyseprozesses (von der Frage bis zur vorläufigen Antwort), der Befähigung von Studierenden zur verantwortungsvollen Nutzung von Daten im beruflichen und akademischen Kontext sowie der Sensibilisierung von Studierenden für reproduzierbare empirische Analysen. Dazu setzen wir in der Lehre auf Unterstützung durch R, RStudio und mosaic.

Das Paket mosaic hilft in der Ausbildung, da es den R Code vereinheitlicht und die Grundlage für ein *Denken mit Daten*<sup>1</sup> bildet. Es findet sich unzählige Literatur zu R und RStudio. Dazu gehört auch [A Student’s Guide to R](#) von Nicholas J. Horton, Randall Pruim und Daniel T. Kaplan, der eine gute Einstiegshilfe für Studierende ist, um erste Schritte in der Datenanalyse mit R zu machen.

Da die englische Sprache ein Hemmnis bei der Einarbeitung in eine neue Materie, wie die der Datenanalyse, sein kann, haben wir uns die Erlaubnis eingeholt, den Student’s Guide auf Deutsch zu übersetzen. Vielen Dank dafür an Nicholas Horton!

Ein solches Projekt kann nur gelingen, wenn sich engagierte Kolleginnen und Kollegen mit einbringen. Daher danken wir an dieser Stelle Roger Bons, Oliver Gansser, Matthias Gehrke, Herbert Hollmann, Tanja Kistler, Andreas Kladroba und Thomas Weiß, die sich der Übersetzung von Kapiteln angenommen haben. Ein besonderer Dank geht an Sebastian Sauer, der ebenfalls Kapitel übersetzt hat und der das Projekt maßgeblich mit angestoßen hat. Ebenso möchten wir uns bei Tabea Griesenbeck, wissenschaftliche

---

<sup>1</sup>Pruim, R., Kaplan, D.T. und Horton, N.J. (2017): The mosaic Package: Helping Students to ‘Think with Data’ Using R. The R Journal, 9(1), 77-10

Mitarbeiterin am ifes, bedanken, die die Organisation, das Layout und die Korrekturen übernommen hat.

Wir hoffen, dass die deutschsprachige Version des Student's Guide to R unseren Studierenden ab dem Wintersemester 2021 eine weitere, zielführende Hilfestellung ist.

Prof. Dr. Bianca Krol

Dekanin | Schlüsselkompetenzen & Methoden Direktorin | ifes Institut für Empirie & Statistik

Prof. Dr. habil. Andrea Schankin (hier fehlt noch der Untertitel und es wäre toll, wenn wir die Namen nebeneinander setzen könnten)

Wo kommt das hier her?: Dieses Buch baut technnisch auf *knitr* [11], *rmarkdown* [1], R [8] und vielen weiteren Open-Source-Anwendungen. Details sind im Kapitel 16 hinterlegt. Lorem Ipsum.

# 1 Einleitung

Dieses Referenzbuch beinhaltet einen Überblick über die Befehle und Funktionen, die zur Bearbeitung von Daten im Rahmen von Statistikkursen für Einsteiger und Fortgeschrittene benötigt werden. Ziel ist es, eine Ergänzung zu den Büchern *Start Teaching with R* [3] und *Start Modeling with R* [5] zur Verfügung zu stellen.

In den meisten der hier verwendeten Beispiele werden Daten der HELP-Studie (Health Evaluation and Linkage to Primary Care) verwendet: Ein randomisiertes klinisches Experiment mit einer neuen Vorgehensweise, die Risikopatienten mit Primärversorgern zusammenbringen soll. Ausführlichere Informationen zu dem Datensatz sind in Kapitel 14 enthalten.

Die Themenauswahl und -reihenfolge ist von Lehrbuch zu Lehrbuch sehr unterschiedlich gestaltet, weswegen die Gliederung dieses Buches anhand der Art der Daten, die analysiert werden sollen, vorgenommen wurde. Damit soll das Nachschlagen und Auffinden benötigter Informationen vereinfacht werden. Gewisse Fähigkeiten im Bereich des Datenmanagements sind für Studierende essentiell [4].

In Kapitel 13 gibt es eine Einführung zu den wichtigsten Kernbegriffen.

Dieses Werk baut auf die Initiativen des MOSAIC Projektes <http://www.mosaic-web.org> auf: Ein durch die amerikanische National Science Foundation (NSF) gefördertes Programm, um den Unterricht von Statistik, Mathematik, Naturwissenschaften und Informatik im Grundstudium zu fördern. Wir werden insbesondere das Paket `mosaic`, das erstellt wurde, um den Einsatz von R in statistischen Kursen zu vereinfachen, und das Paket `mosaicData`, welches eine Reihe von Datensätzen enthält, einsetzen. Das Paket `ggformula` bietet hierbei unter Benutzung der `mosaic`-Syntax Unterstützung für qualitätsvolle Grafiken. Eine Beschreibung des MOSAIC Ansatzes für die Lehre von Statistik und Datenwissenschaft ist verfügbar über <https://journal.r-project.org/archive/2017/RJ-2017-024>. Eine kurze Zusammenfassung der R Befehle, die für die Lehre einführender Statistik essentiell ist, steht in der *vignette* des Paketes `mosaic` zur Verfügung: <https://cran.r-project.org/web/packages/mosaic>.

Auch weitere Ressourcen des MOSAIC Projektes können nützlich sein, zum Beispiel eine kommentierte Sammlung von Beispielen aus verschiedenen Lehrbüchern (vgl. <https://cran.r-project.org/web/packages/mosaic/vignettes/mosaic-resources.html>).

Um ein Paket in R nutzen zu können, muss es zuerst (einmalig) installiert werden und in jeder neuen Session geladen werden. Das Paket `mosaic` kann mit folgendem Befehl installiert werden:

```
install.packages("mosaic")      # Beachten Sie die Anführungszeichen
```

**#** RStudio bietet u. a. auch einen Reiter zur Paketinstallation im Panel rechts unten an.

Das **#** Zeichen stellt einen Kommentar in R dar und klammert den restlichen Text bis zum Ende der Zeile aus.

Nachdem das Paket einmalig installiert wurde, kann es zur Verwendung der enthaltenen Funktionen über folgenden Befehl geladen werden:

```
library(mosaic)
```

R Markdown bietet eine einfache markup Sprache, welche die Ergebnisse in PDF, Word oder HTML übersetzt. Das ermöglicht es, Analysen mit einem Workflow zu versehen, der Reproduzierbarkeit sicherstellt und Kopierfehler vermeidet.



#### Info

Das knitr/LATEX System erlaubt dem erfahrenen Nutzer R und LaTeX im selben Dokument zu verknüpfen. Die Belohnung dafür dieses kompliziertere System zu erlernen ist eine viel feinere Kontrolle über das Ausgabeformat. R Markdown ist jedoch deutlich einfacher zu erlernen und eignet sich auch für fachliche Arbeiten.



#### Hinweis

Um Markdown oder knitr/LATEX nutzen zu können, muss das Paket **markdown** installiert sein.

Die Einführung in R Markdown erfolgt typischerweise zu Beginn des Kurses, begleitet von der Aufforderung dies von vornherein für Seminar- und Hausarbeiten etc. zu nutzen [2].



## 2 Los geht's mit RStudio

RStudio ist eine integrierte Entwicklerumgebung (*integrated development environment* - IDE) für R, die gegenüber anderen eine alternative Schnittstelle zu R mit mehreren Vorteilen bietet:



### Hinweis

Es gibt eine Reihe an Einführungsvideos unter <https://nhorton.people.amherst.edu/rstudio>.

- RStudio läuft auf Mac-, PC-, und Linux-Rechnern und bietet eine vereinfachte Schnittstelle, die vom Aussehen und in der Handhabung auf allen ähnlich ist. Die Standardschnittstellen für R sind ziemlich unterschiedlich auf den verschiedenen Plattformen. Das kann verwirren oder ablenken und zu höherem Unterstützungsaufwand führen.
- RStudio kann in einem Webbrowser ausgeführt werden. Zusätzlich zu eigenständigen Desktop-Versionen oder RStudio.cloud kann RStudio als eine Server-Anwendung installiert werden, die über das Internet zugreifbar ist. Die Web-Schnittstelle ist fast identisch zu der Desktopversion. Wie bei anderen Web-Services auch, melden Nutzer sich an, um Zugriff zu ihrem Konto zu erlangen. Nach Abmeldung und späterer Neuansmeldung, wird die letzte Session wiederhergestellt und Sie können an der Stelle mit ihrer Analyse weiter machen, wo Sie verblieben waren, auch wenn Sie sich auf einem anderen Rechner anmelden. Mit einer etwas fortgeschrittenen Einrichtung können Dozierende die History der R-Session in ihrem Klassenraum speichern und Studierende können diese History-Datei in ihre eigene Umgebung laden.



### Achtung!

Die Desktop- und Server-Versionen von RStudio sind so ähnlich, dass Sie besonders aufpassen müssen, wenn Sie beide gleichzeitig nutzen, um sicher zu gehen, dass Sie in der richtigen Version arbeiten.

# Die Verwendung von RStudio in einem Browser ist wie Facebook für Statistik. Jedes Mal, wenn Anwender sich erneut anmelden, wird die vorherige Session wiederhergestellt und sie können weitermachen, wo sie zuletzt aufgehört haben. Nutzer können sich von jedem Gerät mit Internetzugriff anmelden.

- RStudio ist ein Tool, das die Erstellung reproduzierbarer Forschung unterstützt. Es vereinfacht Text, statistische Analysen (R-Code und R-Output) und grafische Abbildungen zusammen im gleichen Dokument zu vereinen. Das R Markdown-System bietet eine einfache Markup Sprache und fügt die Ergebnisse in HTML zusammen. Das KNITR/LATEX-System versetzt Nutzer in die Lage auch R und LaTeX in dasselbe Dokument zu integrieren. Die Belohnung für das Erlernen dieses komplizierteren Systems ist eine deutlich präzisere Steuerung des Output-Formates. Abhängig von der Niveaustufe des Kurses können Sie eins dieser beiden Systeme für Hausarbeiten und Projekte verwenden.



#### Hinweis

Um Markdown oder KNITR/LATEX zu verwenden, muss das Paket `knitr` auf dem System installiert sein.

- RStudio bietet eine integrierte Unterstützung für das Bearbeiten und Ausführen von R-Code und Dokumenten.
- RStudio bietet einige nützliche Funktionalitäten über eine grafische Benutzeroberfläche (Graphical User Interface oder GUI). RStudio ist nicht ein GUI für R, aber es bietet ein GUI, das Themen wie die Installation und Verwaltung von Paketen, die Steuerung, Speicherung und das Laden von Umgebungen, das Importieren und Exportieren von Daten sowie den Zugriff auf und Export von Abbildungen, Dateien und Dokumentationen vereinfacht.
- RStudio bietet Zugriff auf das Paket `manipulate`. Das Paket `manipulate` bietet eine Möglichkeit, einfach und schnell interaktive grafische Anwendungen zu erstellen.

Obwohl man sicherlich R ohne die Verwendung von RStudio nutzen kann, vereinfacht RStudio viele Dinge und wir empfehlen den Einsatz von RStudio stark. Außerdem erwarten wir in Zukunft noch weitere nützliche Funktionalitäten, da RStudio stetig weiterentwickelt wird.

Wir verwenden in erster Linie eine Online-Version von RStudio. Die Anwendung über den Browser hat den Vorteil, dass Sie nichts installieren oder konfigurieren müssen. Einfach anmelden und Sie können loslegen. Außerdem „merkt“ RStudio sich, was Sie machen und Sie können da weitermachen, wo Sie aufgehört haben, jedesmal wenn Sie sich anmelden (auch auf einem anderen Rechner). Das ist „R in the cloud“ und funktioniert ähnlich wie GoogleDocs oder Facebook für R.

**Hinweis**

R kann von <http://cran.r-project.org/> heruntergeladen und lokal installiert werden. Das Herunterladen und die Installation sind i. A. unkompliziert für Mac-, PC-, oder Linux-Systeme. RStudio ist über <http://www.rstudio.org/> verfügbar.

## 2.1 Mit einem RStudio-Server verbinden

RStudio Server wurden an Hochschulen eingerichtet, um cloud-basierte Berechnungen zu ermöglichen.



RStudio Server wurden schon an vielen Institutionen installiert. Nähere Informationen zu (gebührenfreien) akademischen Lizenzen für RStudio Server Pro und Installationsanweisungen sind über <http://www.rstudio.com/resources/faqs> unter dem ACADEMIC Reiter verfügbar. Der RStudio Server funktioniert mit dem Internet Explorer allerdings nicht sehr zuverlässig.

Sobald Sie mit dem Server verbunden sind, sollten Sie ein Anmeldefenster sehen:

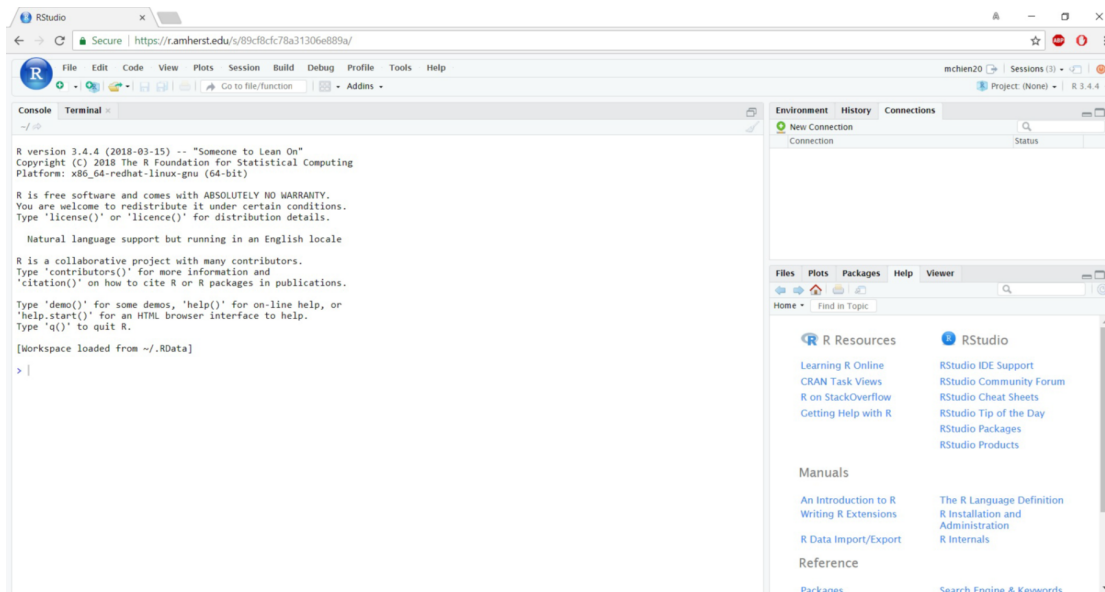
Sign in to RStudio

Username:

Password:

☐ Stay signed in

Wenn Sie sich angemeldet haben, sollten Sie die RStudio Schnittstelle sehen:



Sie können feststellen, dass RStudio seine Welt in vier *Panel* aufteilt. Verschiedene *Panel* sind weiter unterteilt in mehrere Reiter. Welche Reiter in welchem *Panel* auftauchen kann vom Nutzer konfiguriert werden.

R kann vieles mehr als ein einfacher Taschenrechner und wir werden zu gegebener Zeit zusätzliche Eigenschaften vorstellen. Aber, das Durchführen von einfachen Berechnungen in R ist eine gute Vorgehensweise, um die Eigenschaften von RStudio kennenzulernen.

Befehle, die in dem Reiter CONSOLE eingetragen werden, werden direkt ausgeführt von R. Ein guter Start, sich mit der Konsole vertraut zu machen, ist einfache Berechnungen ähnlich wie mit einem Taschenrechner auszuführen. Das meiste funktioniert analog zum typischen Taschenrechner.

Geben Sie folgende Befehle in der Konsole ein:

```
5 + 3
```

```
[1] 8
```

```
15.3 * 23.4
```

```
[1] 358.02
```

```
sqrt(16)      # Quadratwurzel
```

```
[1] 4
```

Dieses letzte Beispiel zeigt, wie Funktionen in R aufgerufen werden und wie Kommentare verwendet werden. Das **#**-Zeichen muss einem Kommentar vorhergehen. Kommentare können sehr nützlich sein bei Skripten mit mehreren Befehlen oder um Code zu erklären.

Werte können zur späteren Weiterverarbeitung mit Variablennamen abgespeichert werden.

**#** Es ist wahrscheinlich am sinnvollsten, sich auf eine Vorgehensweise zwischen rechts-nach-links Zuweisung oder umgekehrt festzulegen, statt hin und her zu wechseln. Wir bevorzugen den Pfeiloperator, weil es visuell darstellt, was in einer Zuweisung passiert und weil es eine klare Unterscheidung zum Zuweisungsoperator darstellt, die Verwendung von `=` weist Variablen Werte zu und die Verwendung von `==` testet auf Gleichheit von Werten.

```
product = 15.3 * 23.4      # Speichere das Ergebnis
product                    # Zeige das Ergebnis an
```

```
[1] 358.02
```

```
product <- 15.3 * 23.4     # <- kann verwendet werden statt =
product
```

```
[1] 358.02
```

Sobald Variablen definiert sind, können sie in anderen Operationen und Funktionen verwendet werden.

```
0.5 * product              # Die Hälfte von product
```

```
[1] 179.01
```

```
log(product)           # (natürlicher) Logarithmus von product
```

```
[1] 5.880589
```

```
log10(product)         # Logarithmus zur Basis 10 von product
```

```
[1] 2.553907
```

```
log2(product)          # Logarithmus zur Basis 2 von product
```

```
[1] 8.483896
```

```
log(product, base = 2) # Logarithmus zur Basis 2 von product, mit expliziter Angabe
```

```
[1] 8.483896
```

Das Semikolon kann verwendet werden, um mehrere Befehle in einer Zeile zu schreiben. Eine übliche Anwendung davon ist es, das Zuweisen eines Ergebnisses und Anzeigen desselben auf einmal zu machen:

```
product <- 15.3 * 23.4; product    # Speichere Ergebnis und zeige es an
```

```
[1] 358.02
```

### 2.1.1 Information zur Version

Manchmal kann es nützlich sein zu prüfen, welche Versionen des Paketes `mosaic`, von R und RStudio Sie verwenden. Die Eingabe von `sessionInfo()` zeigt Informationen über die Version von R und die geladenen Pakete an, während `RStudio.Version()` die Version von RStudio ausgibt.

```
sessionInfo()
```

```
R version 4.1.0 (2021-05-18)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 18363)
```

```
Matrix products: default
```

```
locale:
```

```
[1] LC_COLLATE=German_Germany.1252 LC_CTYPE=German_Germany.1252
[3] LC_MONETARY=German_Germany.1252 LC_NUMERIC=C
[5] LC_TIME=German_Germany.1252
```

```
attached base packages:
```

```
[1] stats      graphics  grDevices  utils      datasets  methods    base
```

```
other attached packages:
```

```
[1] kableExtra_1.3.4  mosaic_1.8.3      ggribes_0.5.3      mosaicData_0.20.2
[5] ggformula_0.10.1  ggstance_0.3.5    dplyr_1.0.7         Matrix_1.3-4
[9] ggplot2_3.3.5     lattice_0.20-44   knitr_1.33
```

```
loaded via a namespace (and not attached):
```

```
[1] ggrepel_0.9.1      Rcpp_1.0.6         svglite_2.0.0       tidyr_1.1.3
[5] assertthat_0.2.1   digest_0.6.27       utf8_1.2.1          ggforce_0.3.3
[9] R6_2.5.0            plyr_1.8.6          backports_1.2.1     labelled_2.8.0
[13] evaluate_0.14      httr_1.4.2          pillar_1.6.1        rlang_0.4.11
[17] rstudioapi_0.13    rmarkdown_2.9       splines_4.1.0       webshot_0.5.2
[21] readr_1.4.0         stringr_1.4.0       htmlwidgets_1.5.3   polyclip_1.10-0
[25] munsell_0.5.0       broom_0.7.8         compiler_4.1.0      xfun_0.24
[29] systemfonts_1.0.2  pkgconfig_2.0.3     htmltools_0.5.1.1   tidyselect_1.1.1
[33] tibble_3.1.2        gridExtra_2.3       mosaicCore_0.9.0    bookdown_0.22
[37] viridisLite_0.4.0  fansi_0.5.0         crayon_1.4.1        withr_2.4.2
[41] MASS_7.3-54         grid_4.1.0          gtable_0.3.0        lifecycle_1.0.0
[45] DBI_1.1.1           magrittr_2.0.1       scales_1.1.1        stringi_1.6.2
[49] farver_2.1.0        leaflet_2.0.4.1     xml2_1.3.2          ellipsis_0.3.2
[53] ggdendro_0.1.22     generics_0.1.0      vctrs_0.3.8         tools_4.1.0
[57] forcats_0.5.1       glue_1.4.2          tweenr_1.0.2        purrr_0.3.4
[61] hms_1.1.0           crosstalk_1.1.1     yaml_2.2.1          colorspace_2.0-2
[65] rvest_1.0.0         haven_2.4.1
```

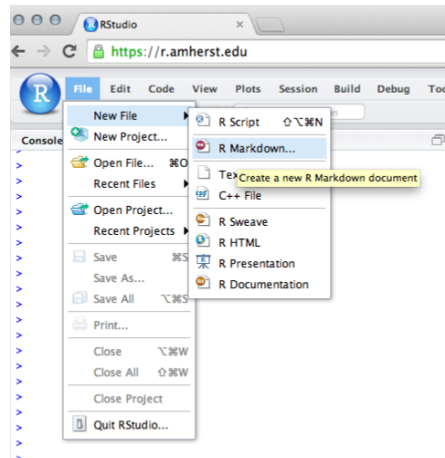
## 2.2 Arbeiten mit Dateien

### 2.2.1 Das Arbeiten mit R-Script Dateien

Eine Alternative ist es, R-Befehle in eine Datei zu speichern. RStudio bietet einen integrierten Editor um diese Dateien zu bearbeiten und unterstützt die Ausführung eines Teils oder aller Befehle. Um eine Datei zu erstellen, gehen Sie in der Menüleiste auf FILE, dann NEW FILE und dann R SCRIPT. Es öffnet sich ein Datei-Editor Fenster im SOURCE Panel. Hier kann R-Code eingetragen werden und es stehen Schaltflächen und Menüpunkte zur Verfügung, um den gesamten Code (das sogenannte *Sourcing* der Datei), einzelne Zeilen oder einen ausgewählten Abschnitt der Datei auszuführen.

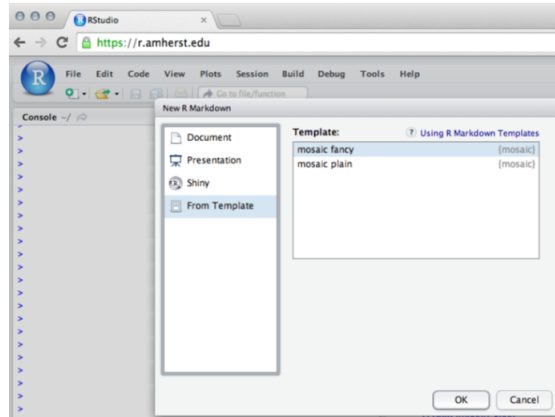
### 2.2.2 Arbeiten mit RMarkdown und knitr/LaTeX

Eine dritte Alternative ist es RStudios Unterstützung für reproduzierbare Forschung zu nutzen. Wenn Sie LaTeX schon kennen, werden Sie die Funktionalitäten des integrierten knitr/LaTeX erforschen wollen. Wenn Sie LaTeX noch nicht kennen, dann bietet das einfachere RMarkdown System eine gute Einführung in die Welt der reproduzierbaren Forschung. Es bietet auch eine gute Möglichkeit für Studierende, ihre Hausarbeiten und Berichte zu erstellen, die Text, R-Code, R-Output sowie Abbildungen enthalten. Um eine neue R Markdown-Datei zu erstellen, wählen Sie FILE, dann NEW FILE, dann R Markdown. Die Datei wird mit einer Kurzvorlage geöffnet, welche die Markup-Sprache skizziert.

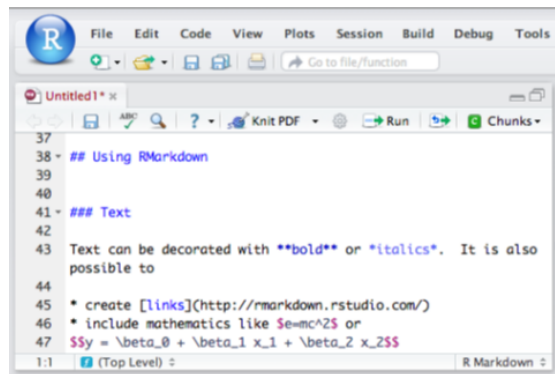


Das Paket `mosaic` enthält zwei nützliche R Markdown Vorlagen für den Einstieg: `fancy` enthält bereits etwas Schnickschnack (und hat zum Ziel, eine Übersicht der Möglichkeiten zu bieten), während `plain` nur ein Grundgerüst enthält und nützlich als Startpunkt für eine neue Analyse ist. Auf diese Vorlagen wird mittels der `TEMPLATE`-Option bei Erstellung einer neuen R Markdown-Datei zugegriffen:

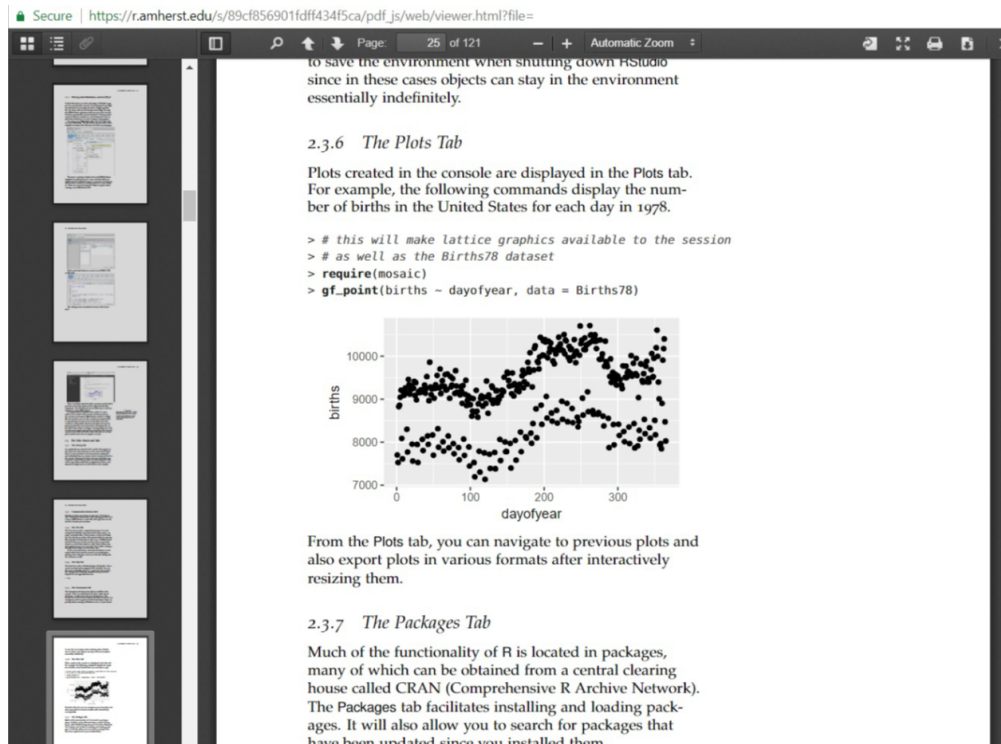




Klicken Sie auf die KNIT-Schaltfläche, um die Markdown-Datei in eine HTML-, PDF-, oder Word-Datei zu konvertieren:



Das erzeugt eine formatierte Version des Dokumentes:



Das Hilfemenü enthält eine “Markdown Quick Reference” und bietet eine kurze Beschreibung der unterstützten Markup Befehle. Die RStudio Webseite enthält weitere ausführlichere Anleitungen über die Verwendung von R Markdown.



### Achtung!

R Markdown und knitr/LaTeX-Dateien haben keinen Zugriff auf die CONSOLE-Umgebung, weshalb der Code in diesen Dateien eigenständig sein muss.

Dass R Markdown- und knitr/LaTeX-Dateien, im Gegensatz zu R-Skripten, die in der CONSOLE ausgeführt werden, keinen Zugriff auf die CONSOLE-Umgebung haben, ist eine gute Eigenschaft, weil die Dateien deshalb eigenständig sein müssen. Dies ermöglicht die Austauschbarkeit und unterstützt gute reproduzierbare Forschungspraktiken. Anfänger, insbesondere wenn sie Analyseschritte in der Konsole ausprobieren und den Code dann über copy & paste in die Datei einfügen, werden zunächst noch häufig Dateien kreieren, die nicht vollständig sind und deshalb nicht korrekt kompilieren.

## 2.3 Die Funktionen der Panel und Reiter

### 2.3.1 Der History-Reiter

Wenn Befehle in der CONSOLE eingetragen werden, dann erscheinen Sie in dem HISTORY-Reiter. Diese Historie kann gespeichert und geladen werden, es gibt eine Suchfunktion, um vorhergehende Befehle zu finden und einzelne Zeilen oder Abschnitte können zurück an die Konsole geschickt werden. Mit geöffnetem HISTORY-Reiter können Sie zurück blättern und auf die vorangegangenen Befehle zugreifen. Das ist vor allem dann nützlich, wenn Befehle viel Output erzeugen und schnell aus dem Bildschirm verschwinden.

### 2.3.2 Kommunikation zwischen Reitern

RStudio bietet verschiedene Möglichkeiten, um R-Code zwischen Reitern auszutauschen. Das Klicken der RUN-Schaltfläche im Editor-Fenster für ein R-Skript oder R Markdown- oder andere Datei kopiert Zeilen mit Code in die Konsole und führt sie aus.

### 2.3.3 Der File-Reiter

Der FILES-Reiter bietet eine einfache Dateiverwaltung an. Sie kann auf übliche Weise bedient und genutzt werden, um Dateien zu öffnen, zu verschieben, umzubenennen oder zu löschen. In der Browser-Version von RStudio bietet der FILES-Reiter auch die Möglichkeit, Dateien hochzuladen, um Dateien vom lokalen Rechner auf den Server zu verschieben. In R Markdown- und knitr-Dateien kann der Code auch in einem bestimmten Abschnitt (CHUNK) oder in allen ausgeführt werden. Jede dieser Eigenschaften ermöglicht es Codes "live" auszuprobieren und gleichzeitig ein Dokument zu erstellen, das eine Aufzeichnung des verwendeten Codes enthält. Umgekehrt kann Code aus der HISTORY zurück in die Konsole kopiert werden, um die Befehle nochmals auszuführen (ggf. nach Bearbeitung) oder in eine Datei im FILE-Reiter eingefügt werden.

### 2.3.4 Der Help-Reiter

In dem HELP-Reiter zeigt RStudio die R-Hilfe-Seiten an. Diese können durchsucht und navigiert werden. Sie können auch eine Hilfeseite öffnen, indem Sie den ?-Operator in der Konsole verwenden. Zum Beispiel führt folgender Befehl dazu, dass die Hilfeseite zur Logarithmusfunktion aufgerufen wird:

```
?log
```

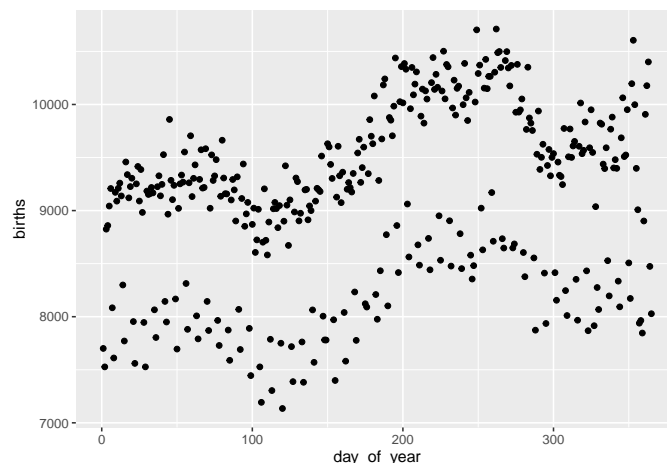
### 2.3.5 Der Environment-Reiter

Der ENVIRONMENT-Reiter zeigt die Objekte, die für die CONSOLE zur Verfügung stehen. Diese sind unterteilt in Daten, Werte (Objekte die weder Dataframe noch Funktionen sind) und Funktionen. Die “Besen”-Schaltfläche kann verwendet werden, um alle Objekte aus der Umgebung zu entfernen und es ist empfehlenswert, das ab und zu durchzuführen, vor allem dann, wenn Sie auf dem RStudio Server arbeiten oder wenn Sie sich dafür entscheiden, die Umgebung beim Schließen von RStudio abzuspeichern. In diesen Fällen könnten die Objekte sonst quasi ewig in der Umgebung gespeichert bleiben.

### 2.3.6 Der Plots-Reiter

Diagramme, die in der Konsole erzeugt werden, werden im PLOTS-Reiter angezeigt. Zum Beispiel zeigen die folgenden Befehle die Anzahl der Geburten in den Vereinigten Staaten für jeden Tag aus 1978 an:

```
library(mosaic)
gf_point(births ~ day_of_year, data = Births78)
```



Innerhalb des PLOTS-Reiters haben Sie Zugriff auf vorherige Grafiken und Sie können diese, nach interaktiver Größenanpassung, in verschiedenen Formaten exportieren.

### 2.3.7 Der Packages-Reiter

Ein großer Teil der R-Funktionalitäten befindet sich in Paketen, die meistens über eine zentrale Anlaufstelle namens CRAN (Comprehensive R Archive Network) verfügbar sind. Der PACKAGES-Reiter vereinfacht die Installation und das Laden der Pakete. Er

ermöglicht es auch Pakete zu finden, die seit ihrer Installation ein Update bekommen haben.

## 3 Eine metrische Variable

### 3.1 Numerische Zusammenfassungen

R beinhaltet eine Reihe von Befehlen, um numerische Variablen zusammenzufassen. Diese beinhalten das Berechnen von Mittelwert, Standardabweichung, Varianz, Median, Interquartilsabstand (IQR) sowie frei wählbare Quantile. Wir werden diese am CESD-Maß (*Center for Epidemiologic Studies-Depression*) für depressive Symptome demonstrieren (diese Werte liegen zwischen 0 und 60; höhere Werte stehen für mehr depressive Symptome). Um die Lesbarkeit der Ausgabe zu verbessern, werden wir die Zahl der ausgegebenen Ziffern begrenzen (s. `?options()` für weitere Konfigurationsmöglichkeiten).

```
library(mosaic)
options(digits = 4)
mean(~cesd, data = HELPrct)
```

```
[1] 32.85
```

Beachten Sie, dass die Funktion `mean()` im Paket `mosaic` eine Formel-Schnittstelle beinhaltet, die `lattice`-Grafiken und linearen Modellen (z. B. `lm()`) ähnlich ist. Das Paket `mosaic` stellt viele weitere Funktionen zur Verfügung, die die gleiche Notation verwenden wie wir in diesem Buch.

#### Tiefer einsteigen

Wenn Sie mit der Notation der Formeln noch nicht vertraut sind, bietet das Begleitbuch *Start Teaching with R* [3] eine detaillierte Präsentation. *Start Modeling with R* [5], ein anderes Begleitbuch, vertieft die Beziehung zwischen dem Modellierungsprozess und der Notation der Formeln.

Dieselbe Ausgabe könnte auch durch die folgenden Befehle erzeugt werden (aber wir werden die MOSAIC-Versionen verwenden, wann immer möglich):

```
with(HELPrct, mean(cesd))
```

```
[1] 32.85
```

```
mean(HELPrct$cesd)
```

```
[1] 32.85
```

Eine ähnliche Funktionalität gibt es auch für andere zusammenfassende Statistiken:

```
sd(~ cesd, data=HELPrct)
```

```
[1] 12.51
```

```
sd(~ cesd, data=HELPrct)^2
```

```
[1] 156.6
```

```
var(~ cesd, data=HELPrct)
```

```
[1] 156.6
```

```
median(~ cesd, data=HELPrct)
```

```
[1] 34
```

Nach dem gleichen Muster lassen sich auch Quantile der Verteilung berechnen:

Standardmäßig stellt die Funktion `quantile()` die Quartile dar, aber ihr kann auch ein Vektor der gewünschten Quantile übergeben werden:

```
with(HELPrct, quantile(cesd))
```

```
0%  25%  50%  75% 100%  
1   25   34   41   60
```

```
with(HELPrct, quantile(cesd, c(.025, .975)))
```

```
2.5% 97.5%
6.3  55.0
```



### Achtung!

Nicht alle Befehle wurden schon für die Nutzung des Formel-Interfaces überarbeitet. Für diese Funktionen muss für den Zugriff auf Variablen innerhalb eines Datensatzes `with()` oder das `$`-Zeichen genutzt werden.

Schließlich stellt die Funktion `favstats()` (für favorite stats) im Paket `mosaic` eine prägnante Zusammenfassung einiger nützlicher Kennzahlen.

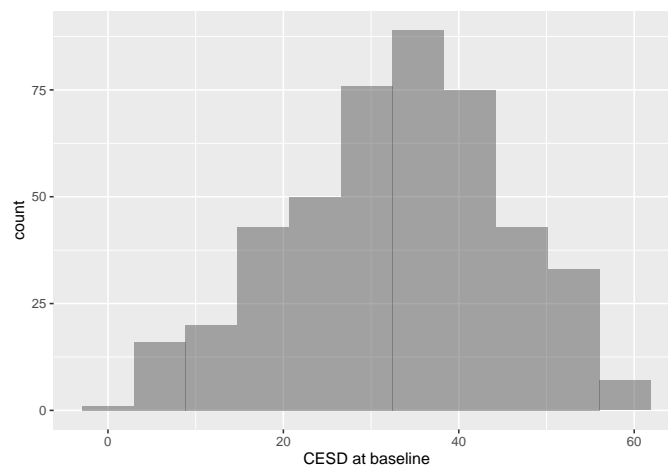
```
favstats(~cesd, data = HELPrct)
```

```
min Q1 median Q3 max  mean    sd   n missing
1  25    34  41   60 32.85 12.51 453      0
```

## 3.2 Grafische Zusammenfassungen

Die Funktion `histogram()` wird genutzt, um ein Histogramm zu erzeugen. Hier nutzen wir das Formel-Interface (wie im Buch *Start Modeling with R* [3]) um ein Histogramm der CESD-Werte zu erzeugen.

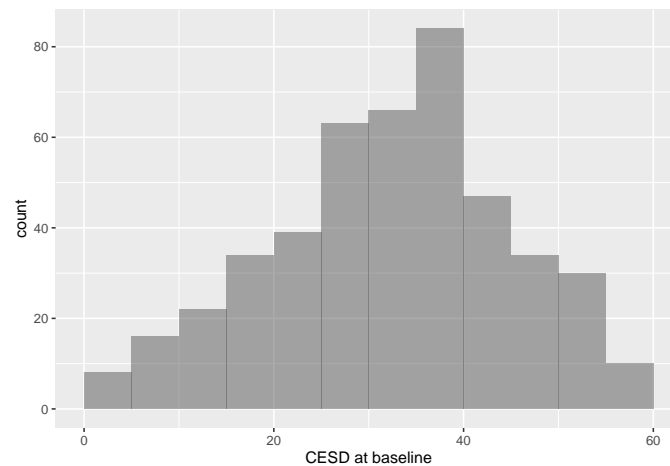
```
gf_histogram(~ cesd, data = HELPrct, binwidth = 5.9)
```





Wir können die Optionen `binwidth()` und `center()` nutzen, um die Lage der Säulen zu steuern.

```
gf_histogram(~ cesd, data = HELPrct, binwidth = 5, center = 2.5)
```



Im Datensatz `HELPrct` ist etwa ein Viertel der Probanden weiblich.

```
tally(~ sex, data = HELPrct)
```

```
sex
female  male
   107    346
```

```
tally(~ sex, format = "percent", data = HELPrct)
```

```
sex
female  male
 23.62  76.38
```

Es ist unkompliziert die Analyse nur auf die weiblichen Probanden zu beschränken. Wenn wir viele Analysen mit einer Teilgruppe unserer Daten durchführen wollen, könnte es am leichtesten sein, einen neuen Datensatz zu erzeugen, der nur die Fälle enthält, die uns interessieren. Die Funktion `filter()` im Paket `dplyr` kann genutzt werden, um einen solchen Datensatz zu erzeugen, der nur die Frauen oder nur die Männer enthält (s. auch Kapitel 13.5). Nachdem dieser erzeugt ist, wird die Funktion `stem()` genutzt, um ein Stamm-Blatt-Diagramm (*stem and leaf plot*) zu erzeugen.

```
Female <- filter(HELPrct, sex == 'female')
Male <- filter(HELPrct, sex == 'male')
with(Female, stem(cesd))
```

The decimal point is 1 digit(s) to the right of the |

```
0 | 3
0 | 567
1 | 3
1 | 555589999
2 | 123344
2 | 66889999
3 | 0000233334444
3 | 5556666777888899999
4 | 00011112222334
4 | 555666777889
5 | 011122222333444
5 | 67788
6 | 0
```

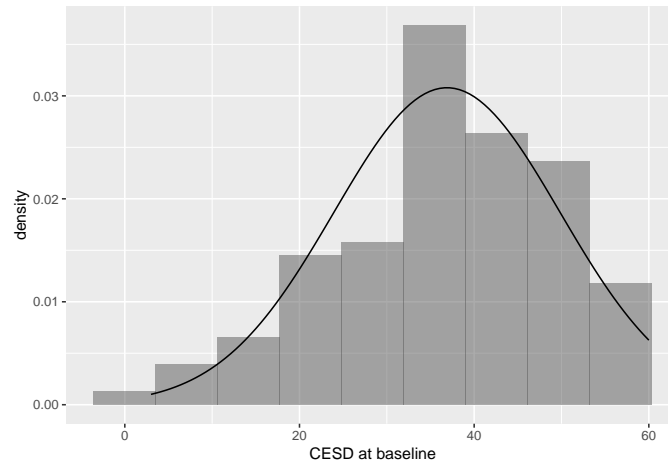


### Achtung!

Um Gleichheit von Ausprägungen zu überprüfen wird ein doppeltes Gleichheitszeichen genutzt!

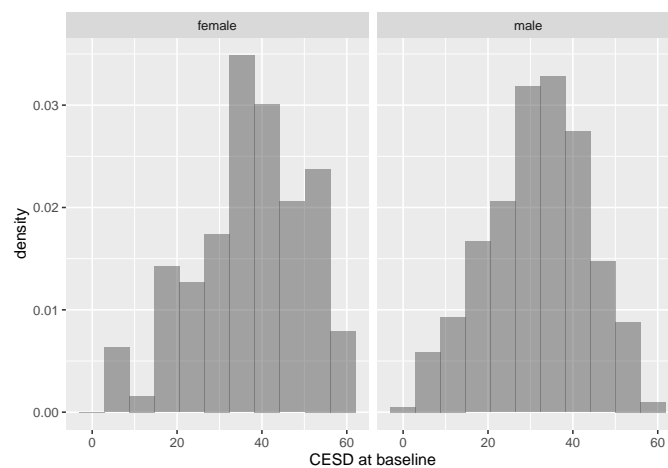
Teilgruppen können auch innerhalb eines Befehls erzeugt werden (dieses Mal mit einer überlagerten Normalverteilung):

```
gf_dhistogram(~ cesd, data = filter(HELPrct, sex == "female"),
  binwidth = 7.1) %>%
  gf_fitdistr(dist = "dnorm")
```



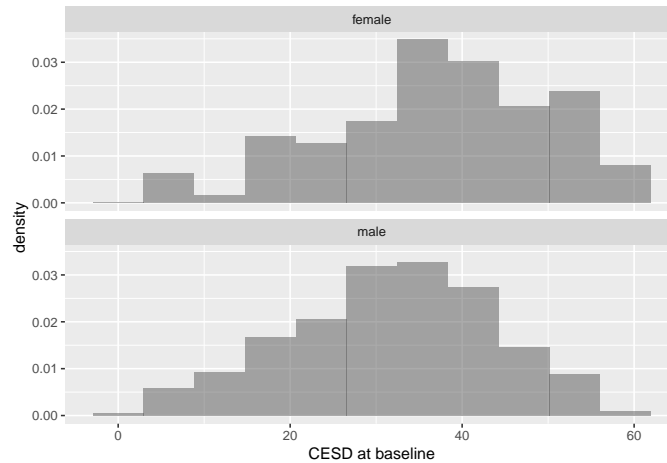
Alternativ können wir nebeneinander angeordnete Plots erzeugen, um mehrere Teilgruppen zu vergleichen.

```
gf_dhistogram(~ cesd, data = HELPrct, binwidth = 5.9) %>%
gf_facet_wrap(~ sex)
```



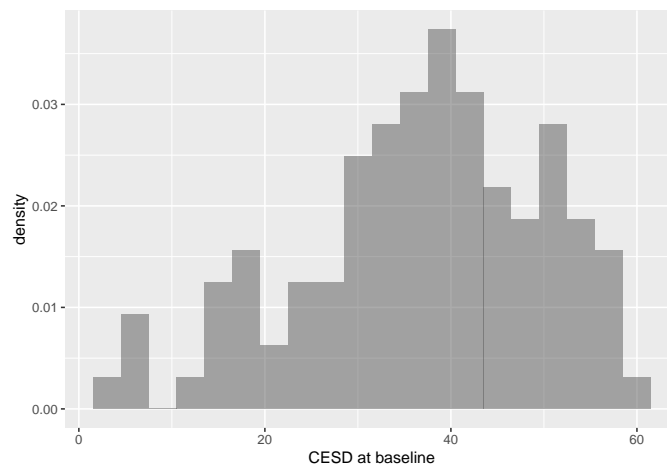
Das Layout kann auch anders angeordnet werden.

```
gf_dhistogram(~ cesd, data = HELPrct, binwidth = 5.9) %>%
gf_facet_wrap(~ sex, nrow = 2)
```



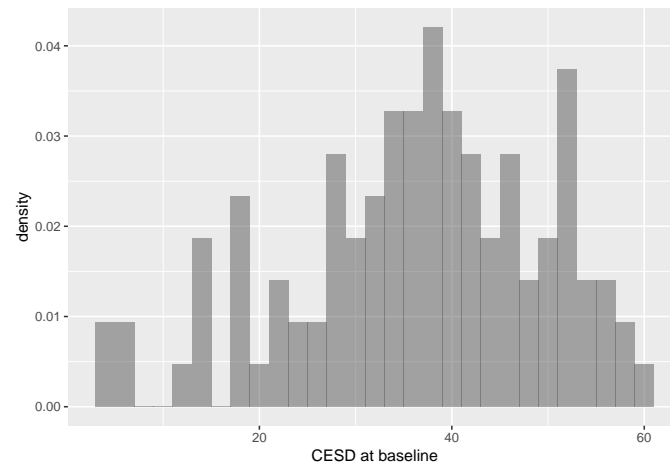
Wir können die Zahl der Säulen auf verschiedene Arten steuern. Hierzu wird die Gesamtzahl angegeben.

```
gf_dhistogram(~ cesd, bins = 20, data = Female)
```



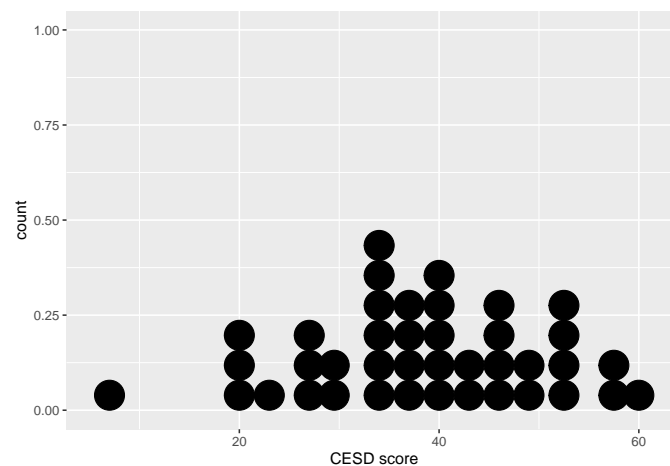
Auch die Breite der Säulen kann vorgegeben werden.

```
gf_dhistogram(~ cesd, binwidth = 2, data = Female)
```



Die Funktion `dotplot()` wird genutzt, um einen *dotplot* für eine kleinere Teilgruppe zu erzeugen (obdachlose Frauen). Wir zeigen auch, wie die Beschriftung der x-Achse geändert werden kann.

```
gf_dotplot(~ cesd, binwidth = 3,
data = filter(HELPrct, sex == "female", homeless == "homeless")) %>%
gf_labs(x = "CESD score")
```



### 3.3 Dichtekurven

Ein Nachteil von Histogrammen besteht darin, dass sie abhängig von der gewählten Anzahl von Säulen sind. Eine andere Darstellung ist die Dichtekurve.

# Dichteplots sind ebenfalls abhängig von diversen Optionen. Wenn Ihr Dichteplot zu zerklüftet oder zu glatt ist, versuchen Sie, das `adjust`-Argument zu variieren: größer als 1 für glattere Plots, kleiner als 1 für gezacktere Plots.

#### Q Tiefer einsteigen

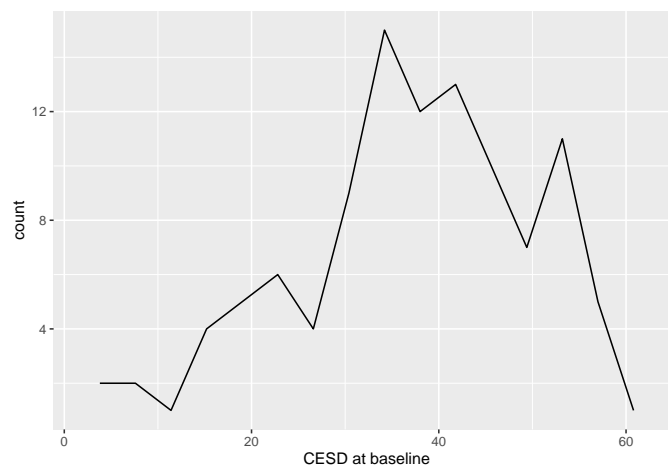
Die Funktion `plotFun()` kann für Anmerkungen an Plots genutzt werden (s. Kapitel 10.2.1).

Hier verzieren wir einen Dichteplot mit einigen Zusätzen, um zu zeigen, wie eine Grafik für didaktische Ziele aufgebaut wird. Wir fügen Text hinzu, legen eine Normalverteilung darüber und ergänzen eine vertikale Linie. Eine Vielzahl von Linienarten, -farben und -stärken stehen zur Auswahl.

## 3.4 Häufigkeitspolygone

Eine dritte Option ist ein Häufigkeitspolygon, bei dem die Grafik erzeugt wird, indem die Mittelpunkte der Säulen eines Histogramms miteinander verbunden werden.

```
gf_freqpoly(~ cesd, data = Female, binwidth = 3.8)
```

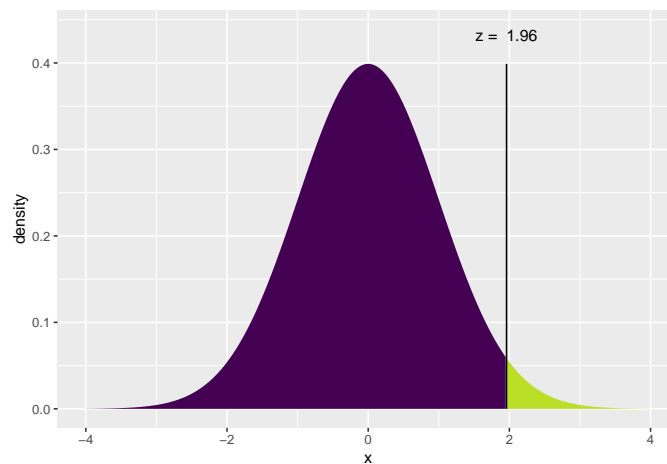


## 3.5 Normalverteilungen

Die berühmteste Dichtekurve ist die Normalverteilung.

Die Funktion `xpnorm()` erzeugt die Wahrscheinlichkeit, dass eine Zufallsvariable kleiner als das erste Argument ist, bei einer Normalverteilung mit dem Mittelwert als zweitem und der Standardabweichung als drittem Argument. Mehr Informationen über Normalverteilungen sind in Kapitel 12 zu finden.

```
xpnorm(1.96, mean = 0, sd = 1)
```



```
[1] 0.975
```



#### Hinweis

x steht für eXtra (außerhalb).

## 3.6 Inferenz einer einzelnen Stichprobe

Das 95%-Konfidenzintervall für den Mittelwert des CESD-Wertes der Frauen kann mit einem t-Test berechnet werden:

```
t.test( ~ cesd, data = Female)
```

One Sample t-test

data: cesd

t = 29, df = 106, p-value <2e-16

alternative hypothesis: true mean is not equal to 0

```
95 percent confidence interval:
 34.39 39.38
sample estimates:
mean of x
 36.89
```

```
confint(t.test( ~ cesd, data = Female))
```

```
      mean of x lower upper level
1      36.89 34.39 39.38  0.95
```

Aber es ist genauso zielführend, es mit der Bootstrap-Methode zu berechnen. Die Statistik, die wir resampeln wollen, ist der Mittelwert.

#### **Tiefer einsteigen**

Mehr Details und Beispiele finden sich in der Vignette zum Resampling im Paket *mosaic*.

```
mean( ~ cesd, data = Female)
```

```
[1] 36.89
```

Ein erstes Resampeln führt zu diesem Ergebnis

```
mean( ~ cesd, data = resample(Female))
```

```
[1] 36.46
```



#### **Hinweis**

Hier resampeln wir durch Ersetzen des Original-Datensatzes, indem wir einen Pseudo-Zufalls-Datensatz mit der gleichen Anzahl von Zeilen erzeugen.



Auch wenn eine einzige Stichprobe von geringem Nutzen ist, so ist es doch sinnvoll, die Studierenden die Berechnung durchführen zu lassen, um zu zeigen, dass sie (normalerweise!) ein anderes Ergebnis erhalten als ohne Resampling.



Eine weitere Durchführung wird zu einem anderen Ergebnis führen:

```
mean( ~ cesd, data = resample(Female))
```

```
[1] 35.23
```

Jetzt resampeln wir 1000 Stichproben und speichern das Ergebnis in einem Objekt namens `trials`:

```
trials <- do(1000) * mean( ~ cesd, data = resample(Female))  
head(trials, 3)
```

```
      mean  
1 38.14  
2 37.93  
3 35.73
```

```
qdata( ~ mean, c(.025, .975), data = trials)
```

```
      2.5% 97.5%  
34.29 39.22
```

## 4 Eine kategoriale Variable

### 4.1 Analyse kategorialer Daten

Mit Hilfe der Funktion `tally()` können absolute und relative Häufigkeiten für kategoriale Daten angegeben werden.

#### Q Tiefer einsteigen

Im Begleitbuch *Start Teaching with R* [3] wird die Formelschreibweise erklärt, die hier ebenfalls verwendet wird. Siehe ebenfalls dort zum Thema statistische Modellierung.

```
tally( ~ homeless, data = HELPrct)
```

```
homeless
homeless  housed
      209      244
```

```
tally( ~ homeless, margins = TRUE,
      data = HELPrct)
```

```
homeless
homeless  housed  Total
      209      244    453
```

```
tally( ~ homeless, format = "percent",
      data = HELPrct)
```

```
homeless
homeless  housed
    46.14    53.86
```

```
tally( ~ homeless, format = "proportion",
      data = HELPrct)
```

```
homeless
homeless  housed
  0.4614    0.5386
```

## 4.2 Der Binomialtest

Ein exaktes Konfidenzintervall für einen Anteilswert (sowie ein Test der Nullhypothese, dass der Bevölkerungsanteil gleich einem bestimmten Wert – standardmäßig 0.5 – ist) kann durch die Funktion `binom.test()` berechnet werden. Die Standard-Funktion `binom.test()` benötigt folgendes Eingabeformat:

```
binom.test(209, 209 + 244)
```

```
data: 209 out of 453
number of successes = 209, number of trials = 453, p-value = 0.1
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.4147 0.5085
sample estimates:
probability of success
      0.4614
```

Mit Hilfe des Paketes `mosaic` kann eine Formelschreibweise verwendet werden, die die vorherige tabellarische Aufbereitung der Daten überflüssig macht.

```
result <- binom.test(~ (homeless == "homeless"),
                    data = HELPrct)
result
```

```
data: HELPrct$(homeless == "homeless") [with success = TRUE]
number of successes = 209, number of trials = 453, p-value = 0.1
```

```
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.4147 0.5085
sample estimates:
probability of success
      0.4614
```

Wie bei Befehlen dieser Art üblich, gibt es eine Menge von nützlichen Informationen, die aus der Ausgabe der Funktion ausgelesen werden können.

```
names(result)
```

```
[1] "statistic"  "parameter"  "p.value"    "conf.int"   "estimate"
[6] "null.value" "alternative" "data.name"
```

Spezifische Informationen werden extrahiert, indem man den Operator `$` oder eine entsprechende Funktion zur Extrahierung nutzt. So kann z. B. das Konfidenzintervall oder der p-Wert verwendet werden.

```
result$statistic
```

```
number of successes
      209
```

```
confint(result)
```

```
      probability of success lower upper level
1      0.4614 0.4147 0.5085 0.95
```

```
pval(result)
```

```
p.value
0.1101
```

**Q Tiefer einsteigen**

Die meisten Objekte in R haben eine Art Druckfunktion. Wenn wir ein Ergebnis angezeigt bekommen, stammt die Ausgabe in der Konsole aus dieser Druckfunktion `print(result)`. Durch den expliziten Aufruf von `print(result)` können oft viele zusätzliche Informationen ausgegeben werden. In einigen Situationen, wie z. B. bei Grafiken, bleibt das Objekt unsichtbar, so dass nichts gedruckt wird. Hier werden die zusätzlichen Informationen üblicherweise nicht benötigt. Falls doch, können Sie aber abgerufen werden.

### 4.3 Der Anteilswerttest

In ähnlicher Weise kann ein geschätztes Intervall und ein approximativer Test auf Anteilswerte über die Funktion `prop.test()` ermittelt werden. Die Anzahl der Personen je Kategorie der Variablen `homeless` wird folgendermaßen ausgezählt:

```
tally(~ homeless, data = HELPrct)
```

```
homeless
homeless  housed
      209      244
```

Die Funktion `prop.test()` berechnet die Anteile und gibt die Ergebnisse aus.

```
prop.test(~ (homeless == "homeless"),
          correct = FALSE, data = HELPrct)
```

```
1-sample proportions test without continuity correction
```

```
data:  HELPrct$(homeless == "homeless")  [with success = TRUE]
X-squared = 2.7, df = 1, p-value = 0.1
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.4160 0.5074
sample estimates:
      p
0.4614
```

Hier untersucht `prop.test` die Variable `homeless` in der gleichen Art und Weise wie `tally()`. Die Funktion `prop.test()` kann ebenso wie `binom.test()` auch direkt mit numerischen Angaben umgehen:

```
prop.test(209, 209 + 244, correct = FALSE)
```

1-sample proportions test without continuity correction

```
data: 209 out of +209 out of 209209 out of 244
X-squared = 2.7, df = 1, p-value = 0.1
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.4160 0.5074
sample estimates:
      p
0.4614
```



#### Info

Wir schreiben `homeless == "homeless"`, um eindeutig festzulegen, welchen Anteil wir betrachten möchten. Wir hätten auch `homeless == "housed"` auswählen können.

`prop.test()` berechnet die Chi-Quadrat-Statistik. Die meisten einführenden Statistikbücher nutzen die Z-Statistik. Sie sind mathematisch äquivalent in Bezug auf inferentielle Aussagen.

## 4.4 Anpassungstests

Es gibt eine Vielzahl von Gütemaßen, die mit Hilfe bestimmter Referenz-Verteilungen bestimmt werden können. Für die Daten der HELP-Studie können wir die Nullhypothese testen, dass in jeder Kategorie des Drogenkonsum ein gleich hoher Anteil an Personen in der Population vorliegt.

```
tally(~ substance, format = "percent",
      data = HELPrct)
```

```
substance
alcohol cocaine heroin
 39.07    33.55    27.37
```

```
observed <- tally(~ substance,
                  data = HELPrct)
observed
```

```
substance
alcohol cocaine heroin
      177      152      124
```



### Achtung!

Zusätzlich zur Option `format` gibt es die Option `margins`, um die Randhäufigkeiten in der Tabelle auszugeben. Die Voreinstellung bei `tally` ist `margins = FALSE`. Probiere es aus!

```
p <- c(1/3, 1/3, 1/3) # equivalent to rep(1/3, 3)
chisq.test(observed, p = p)
```

Chi-squared test for given probabilities

```
data: observed
X-squared = 9.3, df = 2, p-value = 0.01
```

```
total <- sum(observed)
total
```

```
[1] 453
```

```
expected <- total*p
expected
```

```
[1] 151 151 151
```

Wir können die  $\chi^2$ -Statistik auch manuell als Funktion der beobachteten und erwarteten Häufigkeiten unter Unabhängigkeit berechnen.

```
chisq <- sum((observed - expected)^2/(expected))
chisq
```

```
[1] 9.311
```

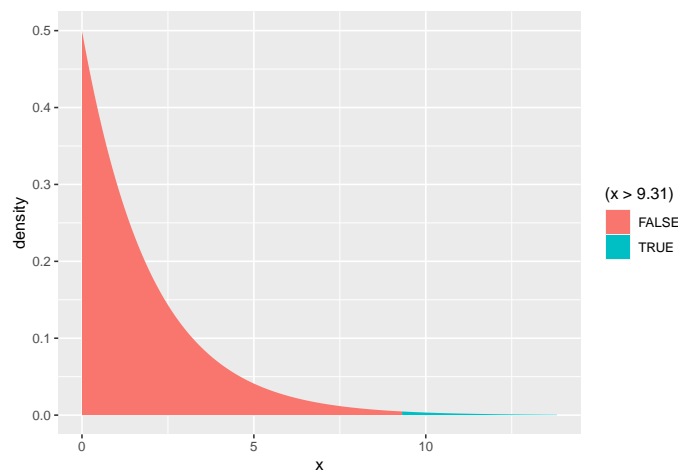
```
1 - pchisq(chisq, df = 2)
```

```
[1] 0.009508
```

# Die Funktion `pchisq()` berechnet die Wahrscheinlichkeit, dass eine  $\chi^2$ -verteilte Zufallsvariable mit `df()` Freiheitsgraden kleiner oder gleich einem gegebenen Wert ist. Hier wird die Gegenwahrscheinlichkeit berechnet, um den Bereich zu finden, der rechts des beobachteten  $\chi^2$ -Wertes liegt.

Es kann hilfreich sein, sich die Verteilung grafisch anzuschauen. Der grün schattierte Bereich zeigt den Bereich rechts vom beobachteten Wert.

```
gf_dist("chisq", df = 2, fill = ~ (x > 9.31),
        geom = "area")
```



Alternativ kann mit Hilfe des Paketes `mosaic` ein analoger `chisq.test()` durchgeführt werden, welcher zusätzlich weitere Angaben liefert, wie z. B. die beobachteten und erwarteten Häufigkeiten.



**Hinweis**

`x` in `xchisq.test()` steht für eXtra.

```
xchisq.test(observed, p = p)
```

Chi-squared test for given probabilities

```
data:  x
X-squared = 9.3, df = 2, p-value = 0.01
```

```
      177      152      124
(151.00) (151.00) (151.00)
[4.4768] [0.0066] [4.8278]
< 2.116> < 0.081> <-2.197>
```

key:

```
observed
(expected)
[contribution to X-squared]
<Pearson residual>
```



Objekte, die sich im Arbeitsspeicher von R befinden, sind unter dem Reiter ENVIRONMENT in RStudio aufgelistet. Diese Liste kann bereinigt werden, indem die nicht mehr benötigten Objekte mit `rm()` gelöscht werden.

```
# nicht mehr benötigte Variablen löschen
rm(observed, p, total, chisq)
```

## 5 Zwei metrische Variablen

### 5.1 Scatterplots

Wir empfehlen den Studierenden immer, jede Analyse durch eine grafische Darstellung ihrer Daten zu beginnen. Hier ergänzen wir ein Streudiagramm des CESD (ein Maß für depressive Beschwerden, höhere Werte deuten auf mehr Beschwerden hin) und des MCS (mental component score des SF-36, wobei höhere Werte auf eine bessere Funktion hinweisen) für weibliche Probanden mit einer LOWESS Linie (locally weighted scatterplot smoother). Die Datenpunkte werden als Kreise dargestellt und die LOWESS-Linie mit einer etwas dickeren Linie dargestellt.

# Die LOWESS-Linie kann dabei helfen, die Linearität eines Zusammenhangs leichter zu erkennen. Sie wird hinzugefügt, indem zwei Punkte und ein LOWESS Filter definiert werden.

```
Female <- filter(HELPrct, female == 1)
gf_point(cesd ~ mcs, data = Female, shape = 1) %>%
  gf_smooth(se = FALSE, size = 2)
```

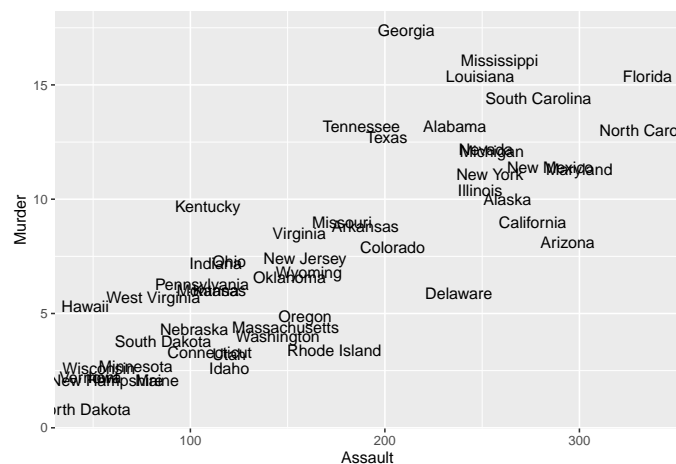


### Q Tiefer einsteigen

Wenn man mit der Ausdrucksweise des statistischen Modellierens nicht vertraut ist, findet man im Begleitbuch *Start Modeling with R* [3] entsprechende Hilfe. Auch in *Start Teaching with R* [5] sind nützliche Tipps für den Einstieg zu finden.

Es ist ganz einfach, etwas anderes als Punkte im Scatterplot zu verwenden. Im folgenden Beispiel wird der Zusammenhang zwischen der Zahl der Überfälle und der Zahl der Morde mit Hilfe der Namen der entsprechenden Bundesstaaten dargestellt.

```
gf_text(Murder ~ Assault,
  label = ~ rownames(USArrests),
  data = USArrests)
```



## 5.2 Korrelationen

Korrelationen können für Variablenpaare und für Variablenmatrizen berechnet werden.

```
cor(cesd ~ mcs, data = Female)
```

```
[1] -0.6738
```

```
smallHELP <- select(Female, cesd, mcs, pcs)
cor(smallHELP)
```

```

      cesd    mcs    pcs
cesd  1.0000 -0.6738 -0.3685
mcs   -0.6738  1.0000  0.2664
pcs   -0.3685  0.2664  1.0000

```

Der Korrelationskoeffizient von Pearson ist voreingestellt. Andere Methoden (z. B. Spearman) können mit der Option `method` gewählt werden.

```
cor(cesd ~ mcs, method = "spearman", data = Female)
```

```
[1] -0.6662
```

### 5.3 Paarweise Plots

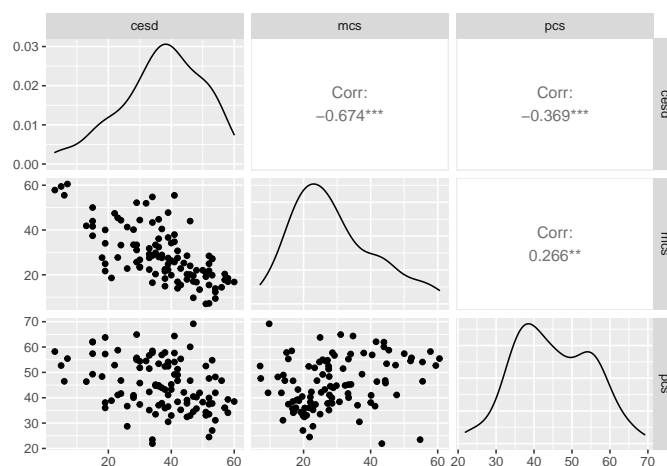
Ein paarweiser Plot (Scatterplot Matrix) kann für jedes Paar eines Variablensets erstellt werden.



#### Hinweis

Das Paket `GGally` unterstützt komplexere Formen von paarweisen Plots.

```
library(GGally)
ggpairs(smallHELP)
```



## 5.4 Einfache lineare Regression

# Normalerweise führen wir die einfache lineare Regression als deskriptive Methode zu einem sehr frühen Zeitpunkt einer Lehrveranstaltung ein.

Lineare Regressionsmodelle werden ausführlich in *Start Modeling with R* [3] beschrieben. Dabei werden die gleichen Befehle für die Spezifizierung von Output und Prädiktoren verwendet, die bereits an früherer Stelle für grafische und numerische Übersichten eingeführt wurden.

Im folgenden betrachten wir das Modell `cesd ~ mcs`.

```
cesdmodel <- lm(cesd ~ mcs, data = Female)
coef(cesdmodel)
```

```
(Intercept)      mcs
      57.349      -0.707
```

# Es ist wichtig, möglichst eindeutige Bezeichnungen für die einzelnen Modellobjekte zu finden. Hier wird der Output von `lm()` als `cesdmodel` gespeichert. Damit wird darauf hingewiesen, dass das Regressionsmodell depressive Beschwerden beschreibt.

Um den Output übersichtlicher zu gestalten schalten wir die Option, die Signifikanzen mit Sternchen zu markieren, ab.

```
options(show.signif.stars = FALSE)
coef(cesdmodel)
```

```
(Intercept)      mcs
      57.349      -0.707
```

```
msummary(cesdmodel)
```

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	57.3485	2.3806	24.09	< 2e-16
mcs	-0.7070	0.0757	-9.34	1.8e-15

Residual standard error: 9.66 on 105 degrees of freedom  
 Multiple R-squared: 0.454, Adjusted R-squared: 0.449  
 F-statistic: 87.3 on 1 and 105 DF, p-value: 1.81e-15

```
coef(summary(cesdmodel))
```

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	57.349	2.38062	24.090	1.425e-44
mcs	-0.707	0.07566	-9.344	1.813e-15

```
confint(cesdmodel)
```

	2.5 %	97.5 %
(Intercept)	52.6282	62.069
mcs	-0.8571	-0.557

```
rsquared(cesdmodel)
```

```
[1] 0.454
```

```
class(cesdmodel)
```

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

Die Ausgabe von `lm()` ist ein Lineares Modell. Ähnlich wie bei `coef()` kann eine Reihe von Operationen auf dieses Objekt zugreifen.

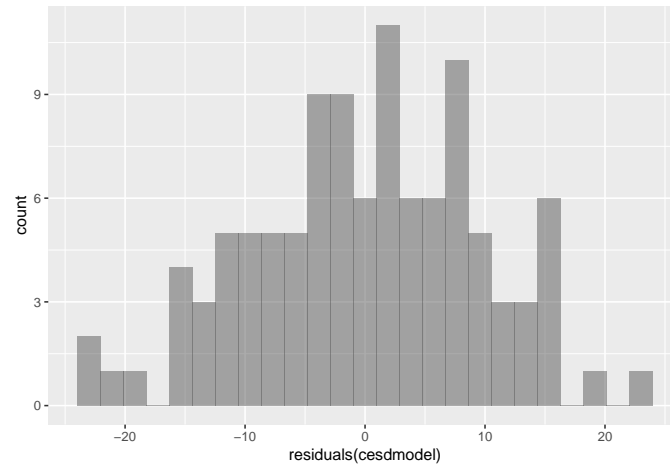
Die Funktion `residuals()` gibt den Residuenvektor aus.



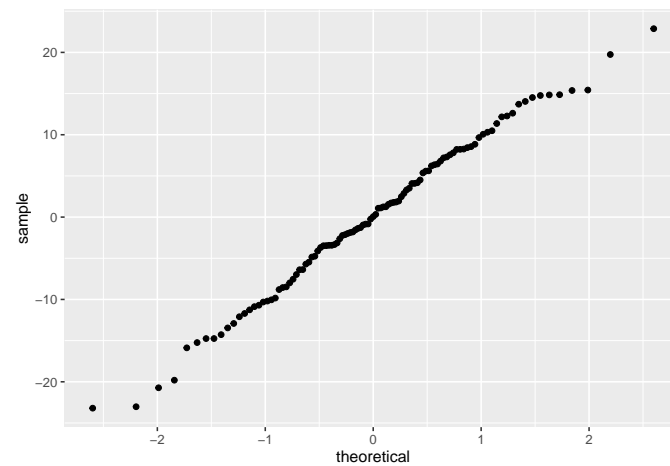
#### Hinweis

Die Funktion `residuals()` kann verkürzt auch als `resid()` geschrieben werden. Eine andere wichtige Funktion ist `fitted()`, die einen Vektor aus geschätzten Werten erzeugt.

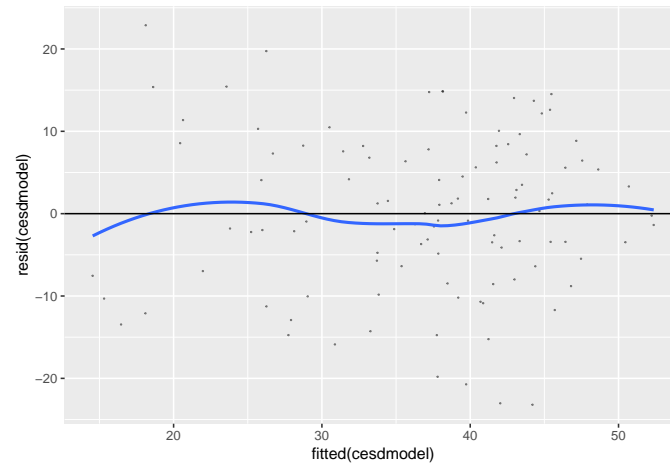
```
gf_histogram(~ residuals(cesdmodel), density = TRUE)
```



```
gf_qq(~ resid(cesdmodel))
```

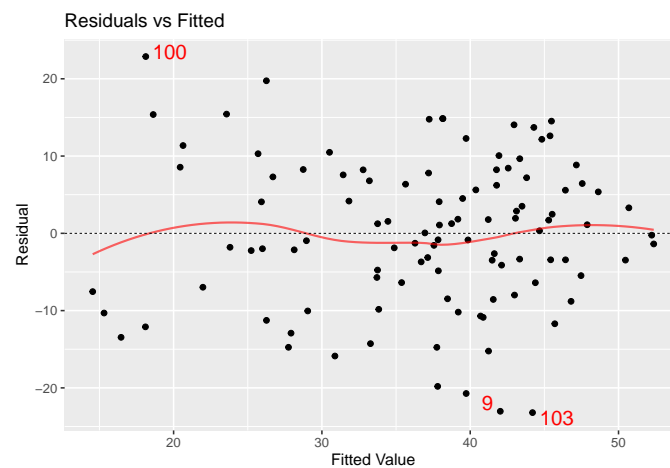


```
gf_point(resid(cesdmodel) ~ fitted(cesdmodel),  
         alpha = 0.5, cex = 0.3, pch = 20) %>%  
  gf_smooth(se = FALSE) %>%  
  gf_hline(yintercept = 0)
```



Mit der Funktion `mplot()` kann eine Reihe nützlicher Plots erzeugt werden. Mit der Option `which = 1` werden geschätzte Werte und Residuen gegenübergestellt:

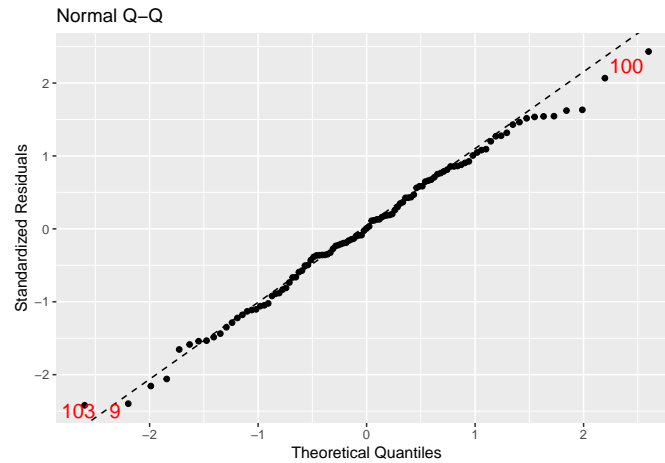
```
mplot(cesdmodel, which = 1)
```



Mit `which = 2` kann ein Quantil-Quantil-Diagramm der Normalverteilung erzeugt werden:

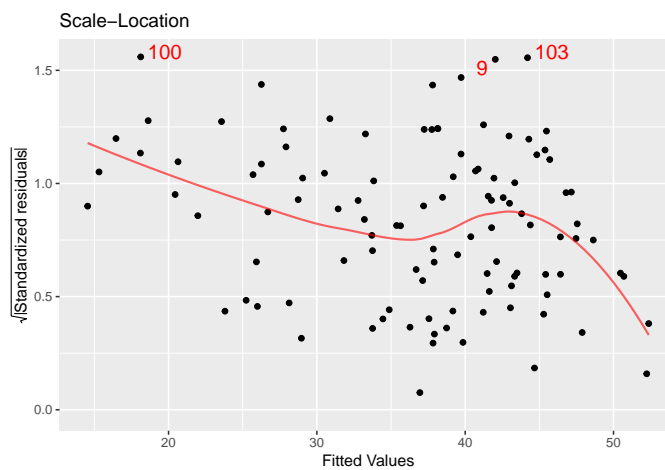
```
mplot(cesdmodel, which = 2)
```





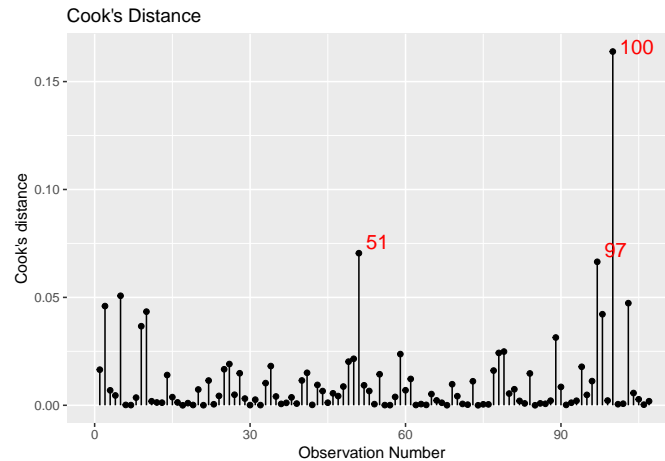
Mit `which = 3` kann ein Scale-Location-Plot erstellt werden:

```
mplot(cesdmodel, which = 3)
```



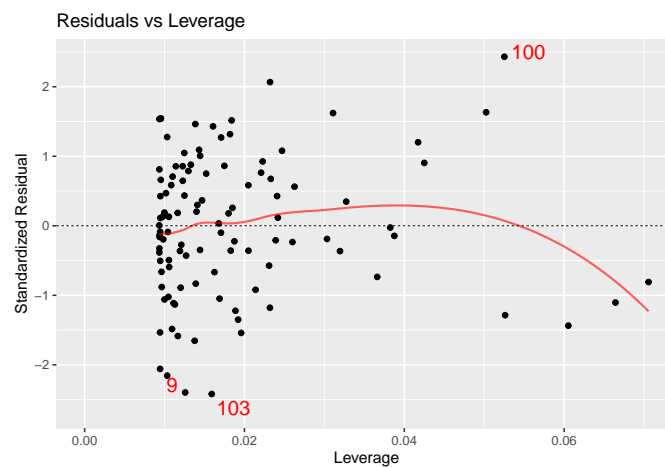
Das Cook's Abstandsmaß in Abhängigkeit von der Beobachtungsnummer wird bei `which = 4` ausgegeben:

```
mplot(cesdmodel, which = 4)
```



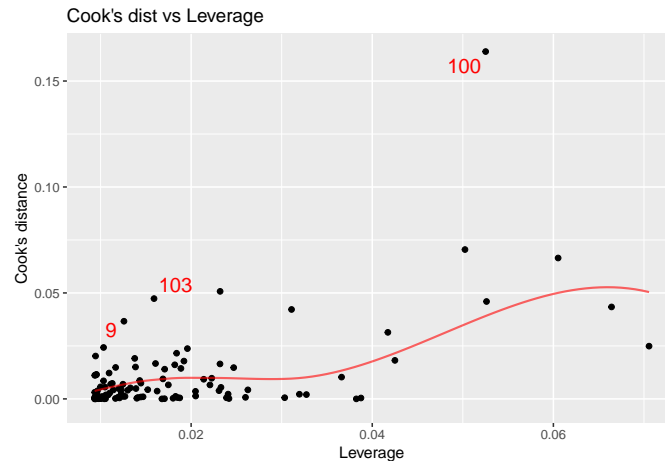
Der Residuen vs. Leverage-Plot, zur Aufdeckung von möglichen einflussreichen Beobachtungen, wird mit `which = 5` erhalten:

```
mplot(cesdmodel, which = 5)
```



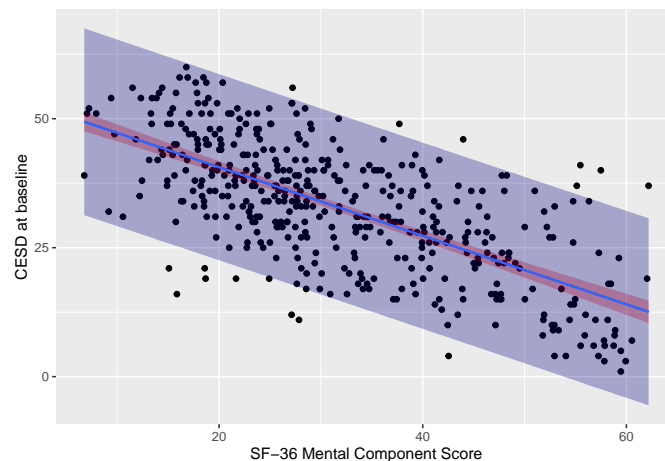
und schlussendlich mit `which = 6` der Cook's distance vs. leverage-Plot:

```
mplot(cesdmodel, which = 6)
```



Vorhersageintervalle können mit Hilfe der Option `interval` in die Funktion `gf_lm()` eingefügt werden.

```
gf_point(cesd ~ mcs, data = HELPrct) %>%
  gf_lm(interval = "confidence", fill = "red") %>%
  gf_lm(interval = "prediction", fill = "navy")
```



### Aufgabe!

Mit Hilfe des Datensatzes `HELPrct` soll eine einfache lineare Regression geschätzt werden, die den Zusammenhang zwischen der am Tag getrunkenen Menge und dem MCS (mental component score) beschreibt. Dieses Modell kann mit Hilfe des Modells `i1 ~ mcs` spezifiziert werden. Interpretieren Sie die Verteilung der Residuen.

## 6 Zwei kategoriale Variablen

### 6.1 Kreuztabellen

Kreuztabellen können für zwei (oder mehr) kategoriale Variablen aufgestellt werden. Hier erstellen wir die Kontingenztafel für den Obdachlosenstatus (obdachlos in einer oder mehr als einer Nacht in den letzten sechs Monaten oder nicht obdachlos) und das Geschlecht.

```
tally(~ homeless + sex, margins = FALSE,
      data = HELPrct)
```

	sex	
homeless	female	male
homeless	40	169
housed	67	177

Wir können auch Spaltenprozentage erzeugen.

```
tally(~ sex | homeless, margins = TRUE,
      format = "percent", data = HELPrct)
```

	homeless	
sex	homeless	housed
female	19.14	27.46
male	80.86	72.54
Total	100.00	100.00

Die Odds Ratios (Chancenverhältnis) können direkt aus der Tabelle berechnet werden.

```
OR <- (40/169)/(67/177)
OR
```

```
[1] 0.6253
```

Das Paket `mosaic` hat eine Funktion, um die Odds Ratios zu berechnen.

```
oddsRatio(tally(~ (homeless == "housed") + sex,
                 margins = FALSE, data = HELPrct))
```

```
[1] 0.6253
```

Die Funktion `CrossTable()` im Paket `gmodels` kann ebenfalls eine Kreuztabelle erzeugen.

```
library(gmodels)
with(HELPrct, CrossTable(homeless, sex,
                         prop.r = FALSE,
                         prop.chisq = FALSE,
                         prop.t = FALSE))
```

```

      Cell Contents
|-----|
|                N |
|      N / Col Total |
|-----|

```

Total Observations in Table: 453

	sex		
homeless	female	male	Row Total
homeless	40	169	209
	0.374	0.488	
housed	67	177	244
	0.626	0.512	
Column Total	107	346	453
	0.236	0.764	

-----|-----|-----|-----|

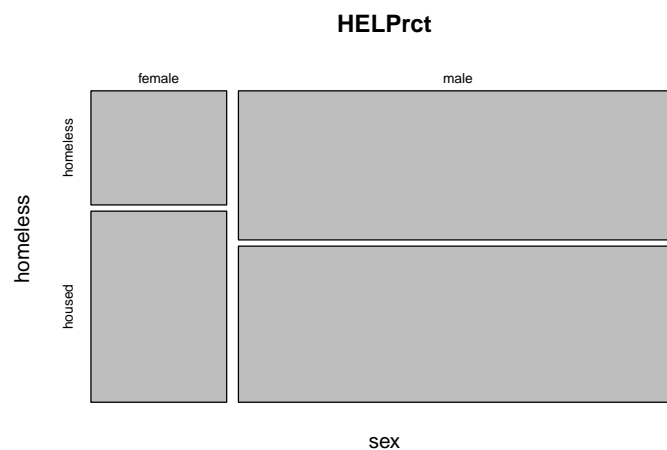
Grafische Zusammenfassungen von Kreuztabellen können ebenfalls nützlich sein. Mosaikplots sind ein Beispiel. Wir sehen, dass Männer in der Gruppe der Obdachlosen überrepräsentiert sind (zu sehen durch die Breite der Fläche, die für Obdachlosen größer ist als für Nicht-Obdachlosen).



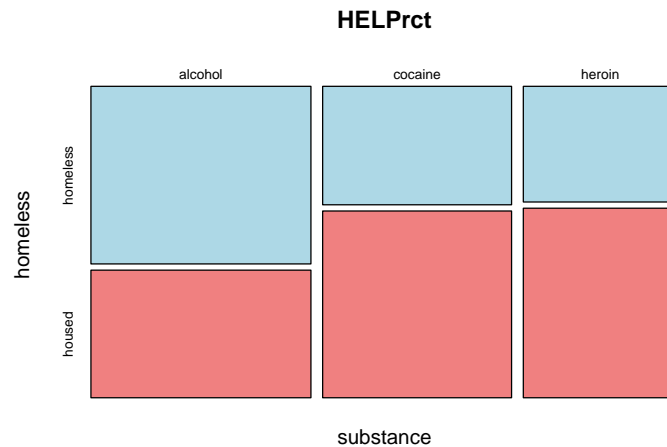
### Achtung!

Die Fachwelt ist immer noch unentschlossen in Bezug auf den Nutzen von Mosaikplots, aufgrund von ihrem geringen data-to-ink-ratio (Daten-Tinten-Verhältnis). Wir haben sie als hilfreich empfunden, um Kontingenztabellen besser zu verstehen [10].

```
mosaicplot(sex ~ homeless, data = HELPrct)
```



```
# farbiges Beispiel
mosaicplot(substance ~ homeless, data = HELPrct,
           color=c("LightBlue", "LightCoral"))
```



## 6.2 Tabellen erstellen

Tabellen können auch aus dem Datensatz selbst erstellt werden. Dazu wird die Funktion `do()` verwendet.

```
HELPrct <- rbind(
  do(40) * data.frame(sex = "female", homeless = "homeless"),
  do(169) * data.frame(sex = "male", homeless = "homeless"),
  do(67) * data.frame(sex = "female", homeless = "housed"),
  do(177) * data.frame(sex = "male", homeless = "housed")
)
tally(~ homeless + sex, data = HELPrct)
```

	sex	
homeless	female	male
homeless	40	169
housed	67	177

## 6.3 Chi-Quadrat-Test

```
chisq.test(tally(~ homeless + sex,
  margins = FALSE,
  data = HELPrct),
  correct = FALSE)
```

## Pearson's Chi-squared test

```
data:  tally(~homeless + sex, margins = FALSE, data = HELPrct)
X-squared = 4.3, df = 1, p-value = 0.04
```

Hier wurde ein statistisch signifikanter Zusammenhang gefunden: Es ist unwahrscheinlich, dass wir einen so starken Zusammenhang finden, wenn der Obdachlosenstatus und das Geschlecht in der Population unabhängig voneinander wären.

Wenn Sie signifikante Zusammenhänge finden, ist es wichtig, dass Sie diese im Kontext des konkreten Problems richtig interpretieren können. Die Funktion `xchisq.test()` liefert weitere Informationen (beobachtete und erwartete Häufigkeiten,  $\chi^2$ -Anteil und Residuen), die hierzu hilfreich sein können.

```
xchisq.test(tally(~ homeless + sex,
                  margins = FALSE,
                  data = HELPrct),
            correct = FALSE)
```

## Pearson's Chi-squared test

```
data:  x
X-squared = 4.3, df = 1, p-value = 0.04
```

```
      40      169
( 49.37) (159.63)
 [1.78]  [0.55]
<-1.33>  < 0.74>
```

```
      67      177
( 57.63) (186.37)
 [1.52]  [0.47]
< 1.23>  <-0.69>
```

```
key:
  observed
  (expected)
 [contribution to X-squared]
 <Pearson residual>
```

Wir beobachten, dass es weniger obdachlose Frauen und mehr obdachlose Männer als erwartet gibt.



## 6.4 Exakter Test nach Fisher

Es kann auch ein exakter Test berechnet werden. Für 2x2-Kontingenztafeln ist dies unkompliziert. Optionen, um die Größe des Problems bei größeren Tabellen zu beschränken, existieren (siehe `?fisher.test()`).

```
fisher.test(tally(~ homeless + sex, margins = FALSE,  
               data = HELPrct))
```

Fisher's Exact Test for Count Data

```
data:  tally(~homeless + sex, margins = FALSE, data = HELPrct)
p-value = 0.05
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 0.3895 0.9968
sample estimates:
odds ratio
 0.6259
```

### Tiefer einsteigen

Beachten Sie die Unterschiede in der Schätzung des Odd Ratios im Vergleich zu Abschnitt 6.1. Die Funktion `fisher.test()` verwendet einen anderen Schätzer (und anderes Intervall, basierend auf dem Profil der Likelihood).

## 7 Metrische Antwortvariable, kategorialer Prädiktor

### 7.1 Eine binäre Variable als Prädiktor: numerische und grafische Zusammenfassung

Hier werden wir die Verteilung der CESD-Scores nach Geschlecht vergleichen. Der Befehl `mean()` kann verwendet werden, um den durchschnittlichen CESD-Score separat für Männer und Frauen zu berechnen.

```
mean(cesd ~ sex, data = HELPrct)
```

```
female  male
 36.89  31.60
```

Der Befehl `favstats()` liefert weitere statistische Kennzahlen pro Gruppe.

```
favstats(cesd ~ sex, data = HELPrct)
```

```
      sex min Q1 median   Q3 max  mean    sd   n missing
1 female  3  29  38.0 46.5  60 36.89 13.02 107         0
2  male   1  24  32.5 40.0  58 31.60 12.10 346         0
```

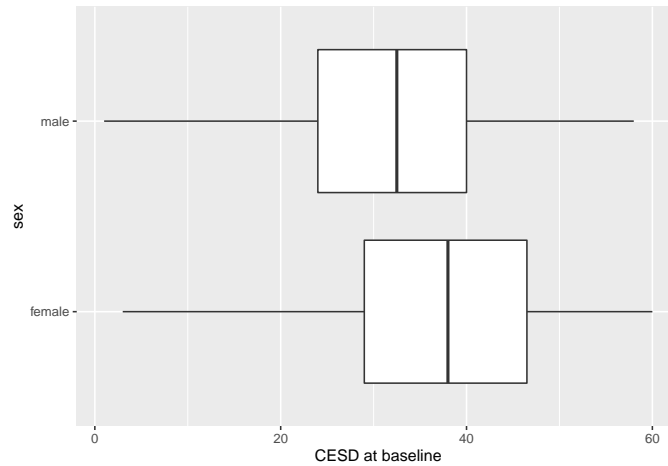
Boxplots sind besonders hilfreiche grafische Darstellungen, um Verteilungen zu vergleichen. Mit dem Befehl `gf_boxplot()` können die Boxplots für die CESD-Scores getrennt nach Geschlecht angezeigt werden. Sowohl aus den numerischen als auch aus den grafischen Zusammenfassungen geht hervor, dass Frauen in der Regel etwas höhere CESD-Werte haben als Männer.



#### Hinweis

Obwohl wir normalerweise die erklärenden Variablen entlang der horizontalen Achse darstellen, so ist die umgedrehte Darstellung für diese Art von Grafen in einigen Fällen vorzuziehen.

```
gf_boxplot(cesd ~ sex, data = HELPrct) %>%
  gf_refine(coord_flip())
```



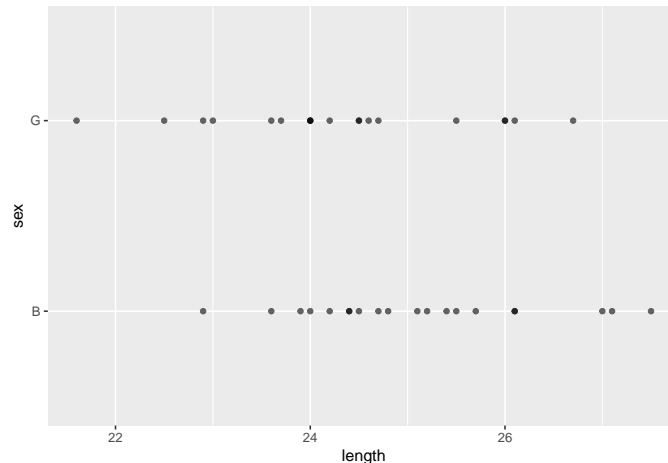
Bei nur kleiner Stichprobengröße gibt es keine Notwendigkeit, diese über Boxplots darzustellen, da der Befehl `gf_point()` ebenfalls kategoriale Prädiktoren darstellen kann. Auch 10–20 Beobachtungen pro Gruppe sind mittels Streudiagramm gut darzustellen. Beobachtungen, die einen gleichen Wert aufweisen, können über die Anpassung des Transparenzlevels (Argument `alpha`) sichtbar gemacht werden.



#### Hinweis

iner von uns sah mal einen Biologen stolz mehrere Boxplots nebeneinander präsentieren. Mit dem Gedanken ein großer Erfolg sei errungen worden, fragte er naiv wie viele Beobachtungen in jeder Gruppe enthalten seien. “Vier”, antwortete der Biologe.

```
gf_point(sex ~ length, alpha = .6, cex = 1.4,
  data = KidsFeet)
```



## 7.2 Ein dichotomer Prädiktor: Zweistichproben-t-Test

Der Students Zweistichproben-t-Test kann ohne (das ist die Voreinstellung; engl. „default“) oder mit der Annahme gleicher Varianzen durchgeführt werden.

```
t.test(cesd ~ sex, var.equal = FALSE, data = HELPrct)
```

Welch Two Sample t-test

```
data: cesd by sex
t = 3.7, df = 167, p-value = 3e-04
alternative hypothesis: true difference in means between group female and group male is not equal to 0
95 percent confidence interval:
 2.493 8.087
sample estimates:
mean in group female    mean in group male
          36.89              31.60
```

Hier sehen wir, dass ein statistisch signifikanter Unterschied zwischen den beiden Gruppen vorliegt.

Wir können den Test unter der Annahme von gleichen Varianzen wiederholen:

```
t.test(cesd ~ sex, var.equal = TRUE, data = HELPrct)
```

## Two Sample t-test

```
data: cesd by sex
t = 3.9, df = 451, p-value = 1e-04
alternative hypothesis: true difference in means between group female and group male is not equal to 0
95 percent confidence interval:
 2.610 7.969
sample estimates:
mean in group female    mean in group male
          36.89              31.60
```

Die Gruppen können ebenso mit dem Befehl `lm()` (ebenfalls unter der Annahme gleicher Varianz) verglichen werden. Der `mosaic`-Befehl `msummary()` gibt eine etwas schlankere Ausgabe im Vergleich zur klassischen Ausgabe des Befehls `summary()` aus.

```
msummary(lm(cesd ~ sex, data = HELPrct))
```

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	36.89	1.19	30.96	< 2e-16
sexmale	-5.29	1.36	-3.88	0.00012

```
Residual standard error: 12.3 on 451 degrees of freedom
Multiple R-squared:  0.0323,    Adjusted R-squared:  0.0302
F-statistic: 15.1 on 1 and 451 DF,  p-value: 0.00012
```

**Hinweis**

Der Befehl `lm()` ist Teil eines sehr viel flexibleren Modellierungsansatzes, wohingegen `t.test()` im Wesentlichen eine nicht weiter anpassungsfähige Sackgasse ist. `lm()` verwendet die Annahme gleicher Varianzen. Siehe hierzu auch das Begleitbuch, *Start Modeling in R* [5] für weitere Details.

## 7.3 Nichtparametrische Zweistichprobentests

Die gleiche Schlussfolgerung wird mit einem nicht-parametrischen Test (Wilcoxon-Rang-Summe) gezogen.

```
wilcox.test(cesd ~ sex, data = HELPrct)
```

Wilcoxon rank sum test with continuity correction

```
data: cesd by sex
W = 23105, p-value = 1e-04
alternative hypothesis: true location shift is not equal to 0
```

## 7.4 Permutationstest

Hier erweitern wir die in Abschnitt 3.6 eingeführten Methoden, um einen zweiseitigen Test einzusetzen, der das Alter zu Beginn getrennt nach Geschlecht vergleicht. Hierzu berechnen wir zunächst die beobachtete Differenz der Mittelwerte:

```
mean(age ~ sex, data = HELPrct)
```

```
female  male
 36.25  35.47
```

```
test.stat <- diffmean(age ~ sex, data = HELPrct)
test.stat
```

```
diffmean
-0.7841
```

Wir können nun die gleiche Statistik erneut berechnen, nachdem wir die Gruppenbezeichnungen zufällig gemischt (permutiert) haben:

```
do(1) * diffmean(age ~ shuffle(sex), data = HELPrct)
```

```
diffmean
1  0.2804
```

```
do(1) * diffmean(age ~ shuffle(sex), data = HELPrct)
```

```
diffmean
1 -0.6373
```

```
do(3) * diffmean(age ~ shuffle(sex), data = HELPrct)
```

```
diffmean
1  0.4150
2 -0.3559
3 -0.2457
```

### Q Tiefer einsteigen

Weitere Details und Beispiele finden Sie in der im Paket `mosaic` enthaltenen `vignette("Resampling")`.

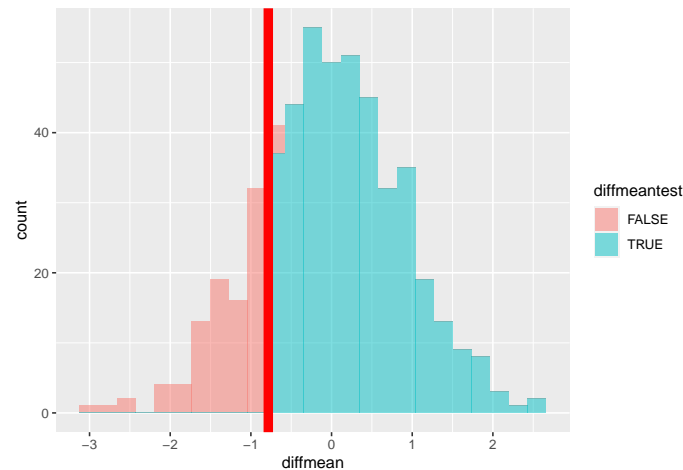
```
rtest.stats <- do(500) * diffmean(age ~ shuffle(sex),
  data = HELPrct)
rtest.stats <- mutate(rtest.stats,
  diffmeantest =
    ifelse(diffmean >= test.stat, TRUE, FALSE))
head(rtest.stats, 3)
```

```
diffmean diffmeantest
1  0.4272          TRUE
2 -1.6651         FALSE
3 -0.6128          TRUE
```

```
favstats(~ diffmean, data = rtest.stats)
```

```
   min    Q1  median    Q3   max   mean    sd  n missing
-2.999 -0.6159 -0.03773 0.5527 2.556 -0.02792 0.8915 500      0
```

```
gf_histogram(~ diffmean, n = 40, xlim = c(-6, 6),
  fill = ~ diffmeantest, pch = 16, cex = .8,
  data = rtest.stats) %>%
  gf_vline(xintercept = ~ test.stat, color = "red", lwd = 3)
```

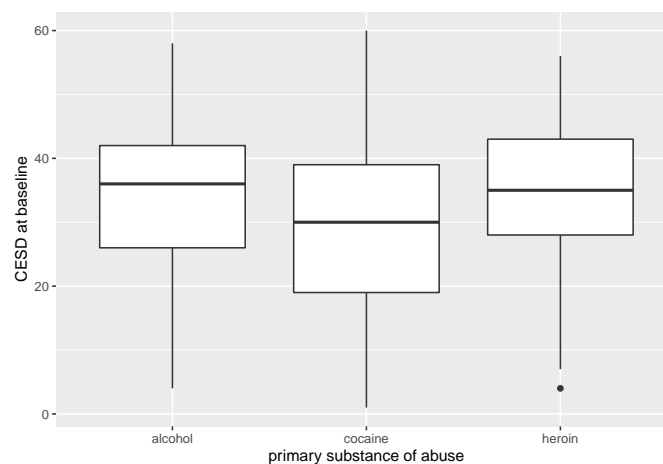


Mit der vorliegenden Permutation erhalten wir keine ausreichenden Belege, um der Nullhypothese, dass Männer und Frauen das gleiche Durchschnittsalter in der Population haben, zu widersprechen.

## 7.5 Einfaktorielle Varianzanalyse

Die vorangegangenen Vergleiche wurden zwischen zwei Gruppen durchgeführt. Wir können ebenso die Mittelwertsvergleiche zwischen drei oder mehr Gruppen mit einer einfachen Varianzanalyse (ANOVA) testen. Hier vergleichen wir den CESD-Score auf Basis des vorwiegenden Drogenkonsums (Heroin, Kokain oder Alkohol), der Median wird im folgenden Diagramm nun mittels einer Linie und nicht mit einem Punkt dargestellt.

```
gf_boxplot(cesd ~ substance, data = HELPrct)
```





```
mean(cesd ~ substance, data = HELPrct)
```

```
alcohol cocaine heroin  
34.37 29.42 34.87
```

```
anovamod <- aov(cesd ~ substance, data = HELPrct)  
summary(anovamod)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
substance	2	2704	1352	8.94	0.00016
Residuals	450	68084	151		

Obwohl die Werte immer noch hoch sind (Scores von 16 oder mehr werden im Allgemeinen als „schwere“ Symptome eingeordnet), weist die Kokain konsumierende Gruppe tendenziell niedrigere Werte auf, als diejenigen, die vorwiegend Alkohol oder Heroin konsumieren.

```
modintercept <- lm(cesd ~ 1, data = HELPrct)  
modsubstance <- lm(cesd ~ substance, data = HELPrct)
```

Der Befehl `anova()` kann mehrere Modelle zusammenfassen.

```
anova(modsubstance)
```

#### Analysis of Variance Table

```
Response: cesd  
          Df Sum Sq Mean Sq F value Pr(>F)  
substance  2  2704    1352    8.94 0.00016  
Residuals 450 68084     151
```

Im vorliegenden Fall sind die Ergebnisse identisch (da es nur einen Prädiktor mit 2 Freiheitsgraden (`df`) gibt).

Der Befehl `anova()` kann auch zum formalen Vergleich zweier (verschachtelter) Modelle verwendet werden.

```
anova(modintercept, modsubstance)
```

Analysis of Variance Table

Model 1: cesd ~ 1

Model 2: cesd ~ substance

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
1	452	70788				
2	450	68084	2	2704	8.94	0.00016

## 7.6 Tukeys Post-hoc-Test

Es gibt eine Vielzahl weiterer Vergleichsverfahren, die nach der Anpassung eines ANOVA-Modells verwendet werden können. Einer davon ist Tukey's Honest Significant Differences (HSD). Weitere Optionen sind innerhalb des Paketes `multcomp` aufgeführt.

```
favstats(cesd ~ substance, data = HELPrct)
```

	substance	min	Q1	median	Q3	max	mean	sd	n	missing
1	alcohol	4	26	36	42	58	34.37	12.05	177	0
2	cocaine	1	19	30	39	60	29.42	13.40	152	0
3	heroin	4	28	35	43	56	34.87	11.20	124	0

```
HELPrct <- mutate(HELPrct,
  subgrp = factor(substance,
    levels = c("alcohol", "cocaine", "heroin"),
    labels = c("A", "C", "H"))
)
mod <- lm(cesd ~ subgrp, data = HELPrct)
HELPHSD <- TukeyHSD(mod, "subgrp")
HELPHSD
```

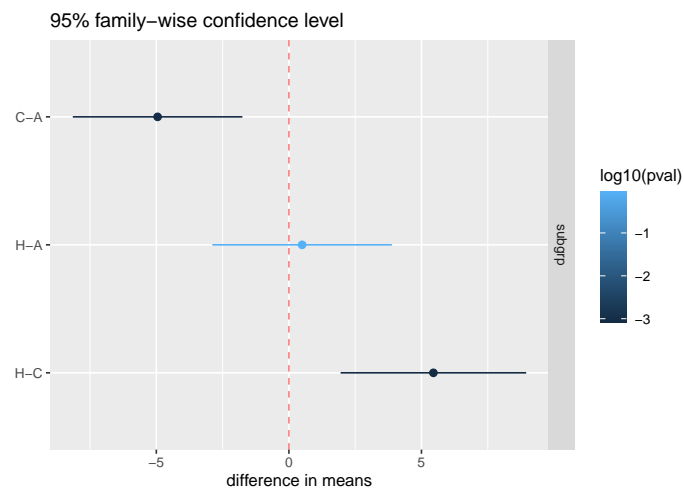
Tukey multiple comparisons of means  
95% family-wise confidence level

Fit: aov(formula = x)

\$subgrp

	diff	lwr	upr	p adj
C-A	-4.9518	-8.150	-1.753	0.0009
H-A	0.4981	-2.889	3.885	0.9362
H-C	5.4499	1.950	8.950	0.0008

```
mplot(HELPHSD)
```



Auch hier zeigt sich, dass die Kokaingruppe deutlich niedrigere CESD-Werte aufweist als eine der beiden anderen Gruppen.

## 8 Kategoriale Antwortvariable, metrischer Prädiktor

### 8.1 Logistische Regression

Die logistische Regression ist mit dem Befehl `glm()` durchführbar, dieser unterstützt eine Vielzahl von Linkfunktionen und Verteilungsformen für generalisierte lineare Modelle, einschließlich logistischer Regression.



#### Hinweis

Der Befehl `glm()` beinhaltet das Argument `family`, mit dem eine Option `link` angegeben werden kann. Der `logit`-Link ist bereits die Voreinstellung für die Binomialverteilung, so dass wir es hier nicht angeben müssen. Die ausführlichere Angabe wäre `family=binomial(link=logit)`.

```
logitmod <- glm(homeless ~ age + female,
                 family = binomial, data = HELPrct)
msummary(logitmod)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	0.8926	0.4537	1.97	0.049
age	-0.0239	0.0124	-1.92	0.055
female	0.4920	0.2282	2.16	0.031

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 625.28 on 452 degrees of freedom  
Residual deviance: 617.19 on 450 degrees of freedom  
AIC: 623.2

Number of Fisher Scoring iterations: 4

```
exp(coef(logitmod))
```

(Intercept)	age	female
2.4415	0.9764	1.6355

```
exp(confint(logitmod))
```

	2.5 %	97.5 %
(Intercept)	1.0081	5.988
age	0.9527	1.000
female	1.0501	2.574

Wir können zwei Modelle vergleichen (für Tests mit mehreren Freiheitsgraden). Beispielsweise könnte uns der Zusammenhang von Obdachlosenstatus und dem Alter je Gruppe des Drogenkonsums interessieren.

```
mymodsubage <- glm((homeless == "homeless") ~ age + substance,
                    family = binomial, data = HELPrct)
mymodage <- glm((homeless == "homeless") ~ age, family = binomial,
                data = HELPrct)
msummary(mymodsubage)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	-0.0509	0.5164	-0.10	0.9215
age	0.0100	0.0129	0.77	0.4399
substancecocaine	-0.7496	0.2303	-3.25	0.0011
substanceheroin	-0.7780	0.2469	-3.15	0.0016

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 625.28 on 452 degrees of freedom  
 Residual deviance: 607.62 on 449 degrees of freedom  
 AIC: 615.6

Number of Fisher Scoring iterations: 4

```
exp(coef(mymodsubage))
```

(Intercept)	age	substancecocaine	substanceheroin
0.9504	1.0101	0.4725	0.4593

```
anova(mymodage, mymodsubage, test = "Chisq")
```

#### Analysis of Deviance Table

Model 1: (homeless == "homeless") ~ age

Model 2: (homeless == "homeless") ~ age + substance

	Resid. Df	Resid. Dev	Df	Deviance	Pr(>Chi)
1	451	622			
2	449	608	2	14.3	0.00078

Wir stellen fest, dass unter Einbeziehung des Alters, die Kokain- und Heroin-Gruppen signifikant seltener obdachlos sind als die Personen, die der Alkoholgruppe zugehörig sind. (Ein ähnliches Ergebnis zeigt sich, wenn nur der Obdachlosenstatus und die Substanz betrachtet werden).

```
tally(~ homeless | substance, format = "percent",
      margins = TRUE, data = HELPrct)
```

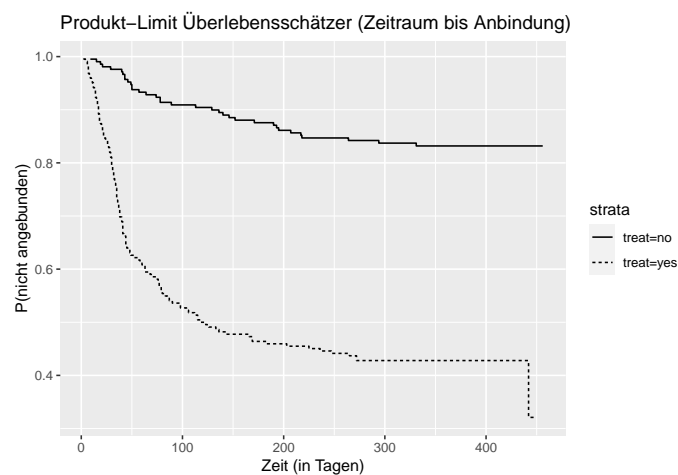
	substance		
homeless	alcohol	cocaine	heroin
homeless	58.19	38.82	37.90
housed	41.81	61.18	62.10
Total	100.00	100.00	100.00

## 9 Überlebenszeitanalysen

Umfangreiche Hilfsmittel für Überlebenszeitanalysen (Zeitraum bis Ereignis) sind aus dem Paket `survival` verfügbar.

### 9.1 Kaplan-Meier-Kurve

```
library(survival)
library(broom)
fit <- survfit(Surv(dayslink, linkstatus) ~ treat,
               data = HELPrct)
fit <- broom::tidy(fit)
gf_step(fit, estimate ~ time, linetype = ~ strata,
        title = "Produkt-Limit Überlebensschätzer (Zeitraum bis Anbindung)",
        xlab = "Zeit (in Tagen)", ylab = "P(nicht angebunden)")
```



Wir sehen, dass die Personen der Behandlungsgruppe (Health Evaluation and Linkage to Primary Care clinic) signifikant häufiger eine Primärversorgung erhalten (eine deutlich geringere Überlebenszeit aufweisen) als die Kontrollgruppe (Grundversorgung).

## 9.2 Cox-Modell mit proportionalem Ausfallrisiko

```
library(survival)
summary(coxph(Surv(dayslink, linkstatus) ~ age + substance,
              data = HELPrct))
```

Call:

```
coxph(formula = Surv(dayslink, linkstatus) ~ age + substance,
      data = HELPrct)
```

```
n= 431, number of events= 163
(22 Beobachtungen als fehlend gelöscht)
```

	coef	exp(coef)	se(coef)	z	Pr(> z )
age	0.00893	1.00897	0.01026	0.87	0.38
substancecocaine	0.18045	1.19775	0.18100	1.00	0.32
substanceheroin	-0.28970	0.74849	0.21725	-1.33	0.18

	exp(coef)	exp(-coef)	lower .95	upper .95
age	1.009	0.991	0.989	1.03
substancecocaine	1.198	0.835	0.840	1.71
substanceheroin	0.748	1.336	0.489	1.15

Concordance= 0.55 (se = 0.023 )

Likelihood ratio test= 6.11 on 3 df, p=0.1

Wald test = 5.84 on 3 df, p=0.1

Score (logrank) test = 5.91 on 3 df, p=0.1

Weder Alter noch Substanzgruppe hatten einen signifikanten Einfluss auf eine Anbindung an Primärversorgung.



# 10 Mehr als zwei Variablen

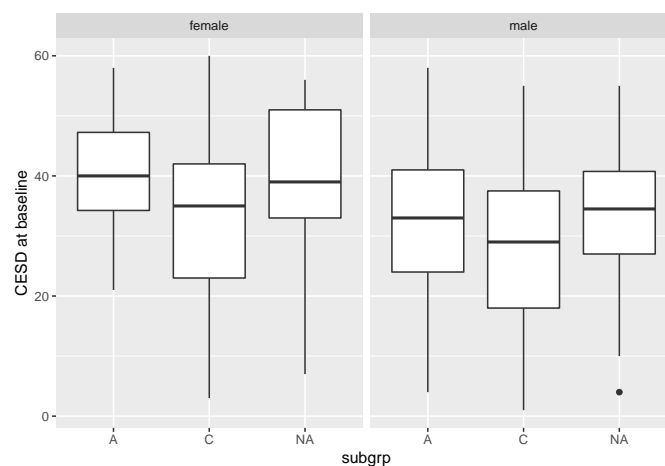
## 10.1 Zwei- (oder mehr-) faktorielle ANOVA

Wir können ein zwei- (oder mehr-) faktorielles ANOVA-Modell, mit oder ohne Interaktion, auf Basis der vereinheitlichten `mosaic`-Syntax erstellen.

```
HELPrct <- mutate(HELPrct,
  subgrp = factor(substance,
    levels = c("alcohol", "cocaine", "heroin"),
    labels = c("A", "C", "H"))
)
median(cesd ~ substance | sex, data = HELPrct)
```

alcohol.female	cocaine.female	heroin.female	alcohol.male	cocaine.male
40.0	35.0	39.0	33.0	29.0
heroin.male	female	male		
34.5	38.0	32.5		

```
gf_boxplot(cesd ~ subgrp | sex, data = HELPrct)
```



```
summary(aov(cesd ~ substance + sex, data=HELPrct))
```

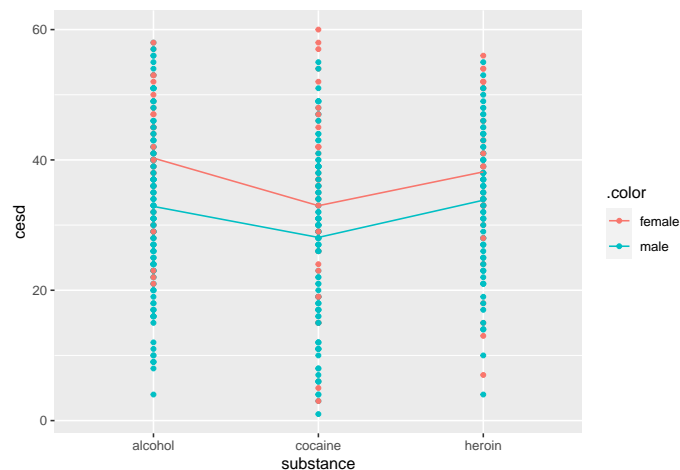
	Df	Sum Sq	Mean Sq	F value	Pr(>F)
substance	2	2704	1352	9.27	0.00011
sex	1	2569	2569	17.61	3.3e-05
Residuals	449	65515	146		

```
summary(aov(cesd ~ substance * sex, data=HELPrct))
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
substance	2	2704	1352	9.25	0.00012
sex	1	2569	2569	17.57	3.3e-05
substance:sex	2	146	73	0.50	0.60752
Residuals	447	65369	146		

Es deutet wenig auf eine Interaktion hin, obwohl signifikante Haupteffekte für die option Substanzgruppen und option sex bestehen.

```
mod <- lm(cesd ~ substance + sex + substance * sex, data = HELPrct)
plotModel(mod)
```



## 10.2 Multiple Regression

Multiple Regression ist eine logische Erweiterung der vorangegangenen Befehle, sobald zusätzliche Prädiktoren hinzukommen. Dies ermöglicht es den Studierenden mit dem Versuch zu beginnen, multivariate Beziehungen zu entflechten.

# Wir neigen dazu, multiple lineare Regression zeitig in unseren Kurse einzuführen, als eine rein deskriptive Methode, und dann immer wieder zu ihr zurückzukehren. Die Beweggründe werden ausführlich im Begleitmaterial *Start Modeling with R* [5] beschrieben

Hier betrachten wir ein Modell (Parallelsteigungen) für die depressive Symptomatik als eine Funktion aus dem Mental Component Score (MCS), dem Alter (in Jahren) und dem Geschlecht der Person.

```
lmnointeract <- lm(cesd~mcs+age+sex,data=HELPrct)
msummary(lmnointeract)
```

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	53.8303	2.3617	22.79	<2e-16
mcs	-0.6548	0.0336	-19.50	<2e-16
age	0.0553	0.0556	1.00	0.3200
sexmale	-2.8993	1.0137	-2.86	0.0044

Residual standard error: 9.09 on 449 degrees of freedom  
 Multiple R-squared: 0.476, Adjusted R-squared: 0.473  
 F-statistic: 136 on 3 and 449 DF, p-value: <2e-16

Wir können ebenso ein Modell anpassen, das eine Interaktion zwischen MCS und `sex` einschließt.

```
lminteract <- lm(cesd ~ mcs + age + sex + mcs:sex, data= HELPrct)
msummary(lminteract)
```

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	55.3906	2.9903	18.52	<2e-16
mcs	-0.7082	0.0712	-9.95	<2e-16
age	0.0549	0.0556	0.99	0.324
sexmale	-4.9421	2.6055	-1.90	0.058
mcs:sexmale	0.0687	0.0807	0.85	0.395

Residual standard error: 9.09 on 448 degrees of freedom  
 Multiple R-squared: 0.477, Adjusted R-squared: 0.472  
 F-statistic: 102 on 4 and 448 DF, p-value: <2e-16

```
anova(lminteract)
```

#### Analysis of Variance Table

Response: cesd

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
mcs	1	32918	32918	398.27	<2e-16
age	1	107	107	1.29	0.2563
sex	1	676	676	8.18	0.0044
mcs:sex	1	60	60	0.72	0.3952
Residuals	448	37028	83		

```
anova(lmnointeract, lminteract)
```

#### Analysis of Variance Table

Model 1: cesd ~ mcs + age + sex

Model 2: cesd ~ mcs + age + sex + mcs:sex

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
1	449	37088				
2	448	37028	1	59.9	0.72	0.4

Es deutet wenig auf einen Interaktionseffekt hin, sodass wir diese aus dem Modell entfernen können.

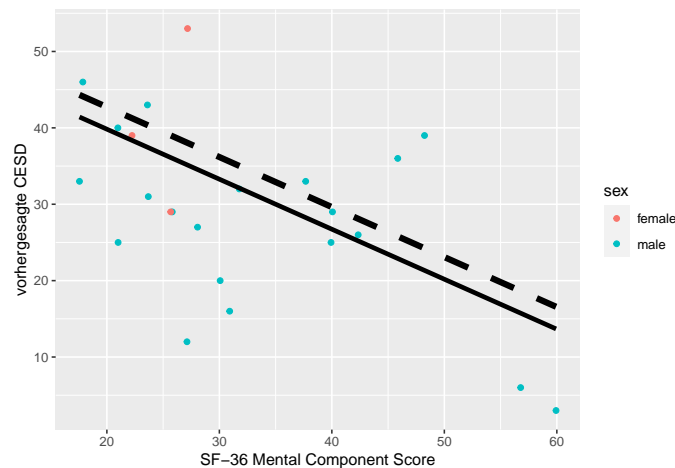
### 10.2.1 Visualisierung der Regressionsergebnisse

Die Funktionen `makeFun()` und `plotFun()` aus dem Paket `mosaic` können zur Abbildung der vorhergesagten Werte des Regressionsmodells verwendet werden. Für dieses Beispiel könnten wir die vorhergesagten CESD-Werte für einen MCS-Wertebereich (mental component score) ausgeben, die eine hypothetisch 36-jährige männliche und weibliche Person in einem Modell mit parallelen Steigungen (keine Interaktion) haben könnte.

```
lmfunction <- makeFun(lmnointeract)
```

Wir können nun die vorhergesagten Werte - getrennt nach männlichen und weiblichen Personen - für einen MCS-Wertebereich (mental component score), zusammen mit den beobachteten Werten aller 36-jähriger darstellen.

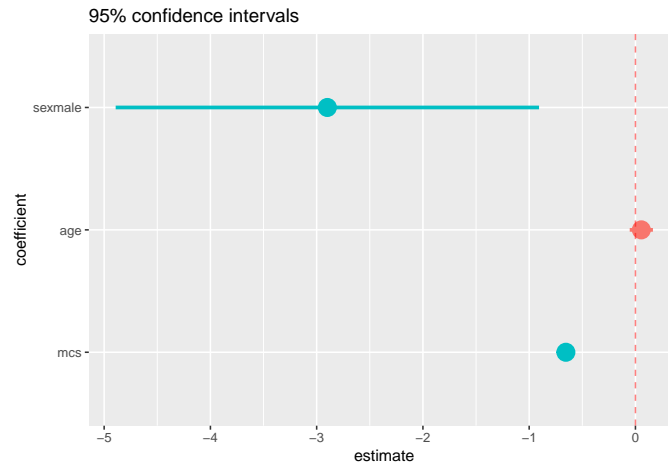
```
gf_point(cesd ~ mcs, color = ~ sex,
         data = filter(HELPrct, age == 36),
         ylab = "vorhergesagte CESD") %>%
gf_fun(lmfunction(mcs, age=36, sex = "male") ~ mcs,
       xlim = c(0,60), size = 1.5) %>%
gf_fun(lmfunction(mcs, age=36, sex = "female") ~ mcs,
       xlim = c(0,60), linetype = 2, size = 2)
```



### 10.2.2 Koeffizientendarstellung

Manchmal ist es nützlich eine Darstellung der Koeffizienten eines multiplen Regressionsmodells (zusammen mit den entsprechenden Konfidenzintervallen) zu erstellen.

```
mplot(lmnointeract, rows = -1, which = 7)
```



# Dunklere bzw. hier die blauen Punkte weisen auf Regressionskoeffizienten hin, bei denen das 95%-Konfidenzintervall nicht den Wert Null der Nullhypothese beinhaltet.



#### Achtung!

Vorsicht bei der Generierung eines Regressionsmodells, bei Vorliegen von fehlenden Werten (siehe Abschnitt 13.11)

### 10.2.3 Auswertung der Residuen

Es ist recht einfach die Residuen des Modells auszuwerten. Wir beginnen damit, die angepassten Werte und die Residuen den Daten hinzuzufügen.



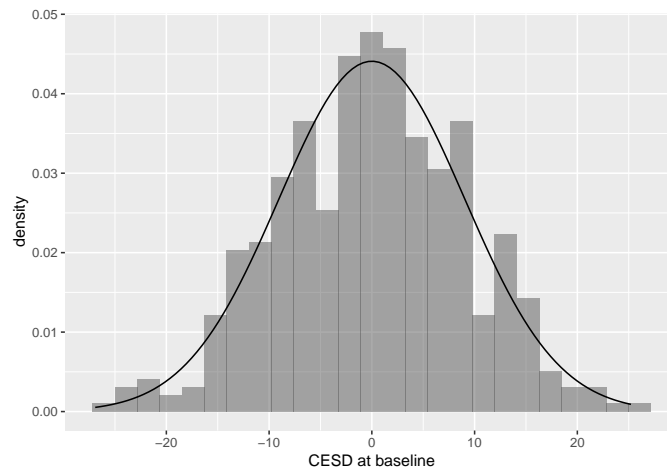
#### Hinweis

ie Funktion `mpplot()` kann ebenfalls zur Erstellung dieser Grafiken verwendet werden.



Hier fügen wir zwei neue Variablen zu einem bestehenden Datensatz hinzu. Es ist häufig eine gute Programmierpraxis, dem daraus resultierenden Datensatz einen neuen Namen zu geben. Für dieses Dokument behalten wir den Namen allerdings der Einfachheit halber bei.

```
HELPrct <- mutate(HELPrct,
  residuals = resid(lmnointeract),
  pred = fitted(lmnointeract))
gf_dhistogram(~ residuals, data= HELPrct) %>%
  gf_fitdistr(dist = "dnorm")
```



Wir können den Teil der Beobachtungen mit extrem hohen Residuen bestimmen.

```
filter(HELPrct, abs(residuals)>25)
```

	age	anysubstatus	anysub	cesd	d1	daysanysub	dayslink	drugrisk	e2b	female	sex
1	43	0	no	16	15	191	414	0	NA	0	male
2	27	NA	<NA>	40	1	NA	365	3	2	0	male

	g1b	homeless	i1	i2	id	indtot	linkstatus	link	mcs	pcs	pss_fr	racegrp
1	no	homeless	24	36	44	41	0	no	15.86	71.39	3	white
2	no	homeless	18	18	420	37	0	no	57.49	37.75	8	white

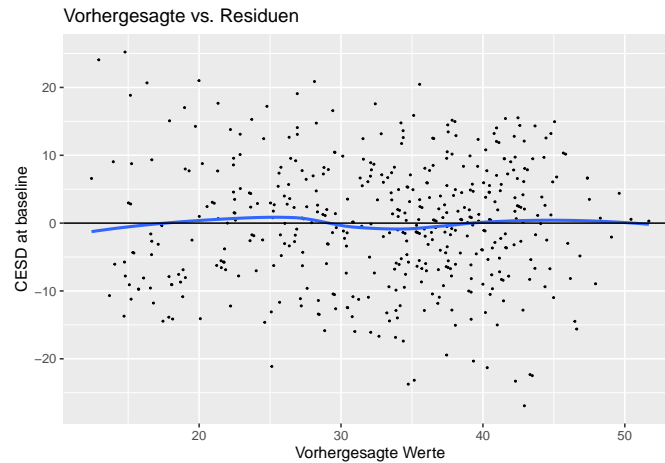
	satreat	sexrisk	substance	treat	avg_drinks	max_drinks	hospitalizations	subgrp
1	no	7	cocaine	yes	24	36	15	C
2	yes	3	heroin	no	18	18	1	<NA>

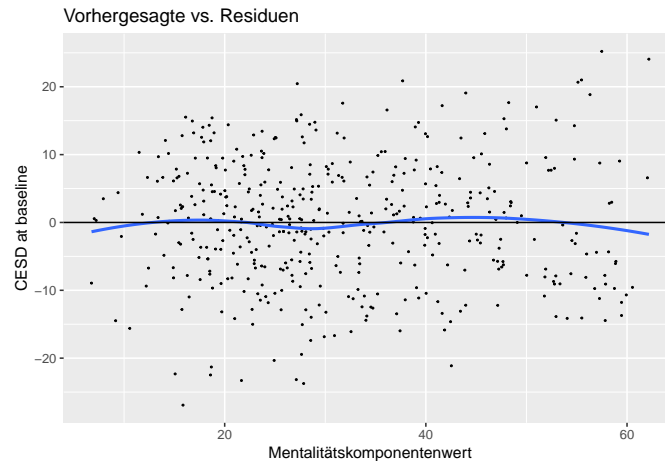
	residuals	pred
1	-26.92	42.92
2	25.22	14.78

```
gf_point(residuals ~ pred, cex = .3, xlab = "Vorhergesagte Werte",
  title = "Vorhergesagte vs. Residuen", data = HELPrct) %>%
```

```
gf_smooth(se = FALSE) %>%
gf_hline(yintercept = 0)
```



```
gf_point(residuals ~ mcs, cex = .3, xlab = "Mentalitätskomponentenwert",
         title = "Vorhergesagte vs. Residuen", data = HELPrct) %>%
gf_smooth(se = FALSE) %>%
gf_hline(yintercept = 0)
```



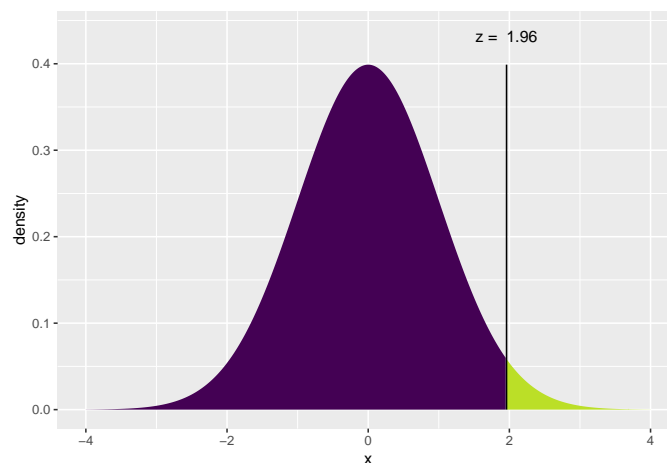
Die Annahme der Normalverteilungs- (s. Histogramm), Linearitäts-, und Heteroskedastizitätsvoraussetzung (s. Residuenplots - jeweils eine horizontale Linie ohne Muster oder Trends in der Punktwolke) erscheint hier vernünftig.



# 11 Wahrscheinlichkeitsverteilungen und Zufallsvariablen

R kann Werte für zahlreiche Verteilungsfunktionen berechnen. Es ist leicht, aus diesen Verteilungsfunktionen Zufallsdaten zu generieren, die dann für Simulationen und explorative Untersuchungen genutzt werden können.

```
xpnorm(1.96, mean = 0, sd = 1) #  $P(Z < 1.96)$ 
```



```
[1] 0.975
```

```
# Wert, der  $P(Z < z) = 0.975$  erfüllt  
qnorm(.975, mean = 0, sd = 1)
```

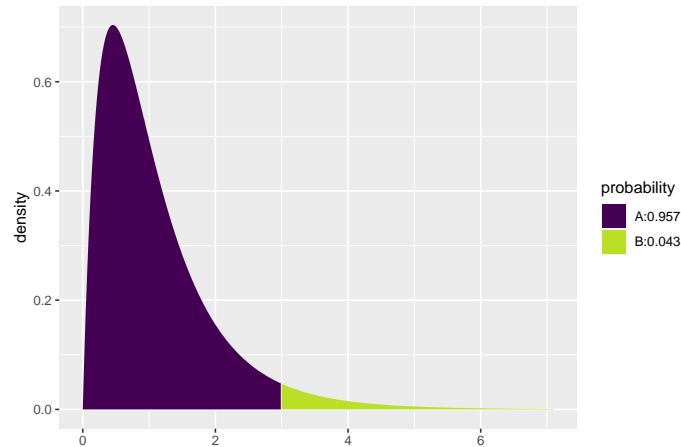
```
[1] 1.96
```

```
integrate(dnorm, -Inf, 0) #  $P(Z < 0)$ 
```

```
0.5 with absolute error < 4.7e-05
```

Eine vergleichbare Ausgabe gibt es für die F-Verteilung.

```
xpf(3, df1 = 4, df2 = 20)
```



```
[1] 0.9568
```

Die folgende Tabelle zeigt die Basisnamen für die Verteilungen, die in R zur Verfügung stehen. Die Vorsilbe **d** erzeugt die Dichtefunktion für die jeweilige Verteilung, **p** die kumulierte Dichte- bzw. Verteilungsfunktion, mit **q** erhalten Sie die Quantile und **r** erzeugt Zufallsziehungen. Um beispielsweise die Dichtefunktion der Exponentialverteilung zu finden, nutzen Sie das Kommando `dexp()`. Die Funktion `qDIST()` ist die inverse Funktion zu `pDIST()` (für einen Basisnamen `DIST`).

```
kable(RVerteilungen, format = "latex", booktabs=T) %>%
  kable_styling(latex_options = c("striped", "HOLD_position", "condensed"),
    full_width = F,
    font_size = 10)
```

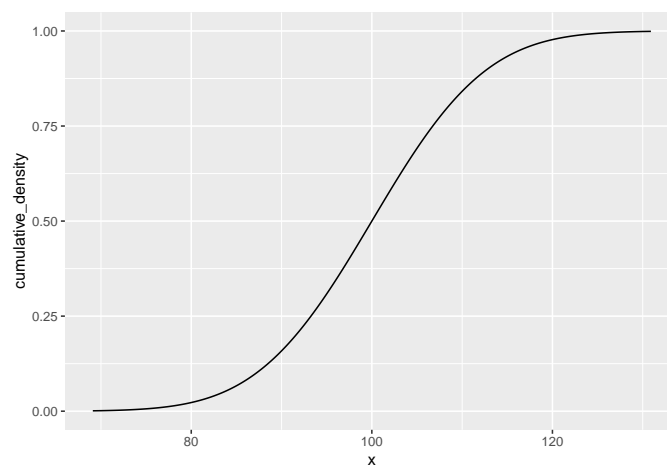
Verteilung	Basisname
Beta	beta
Binomial	binom
Cauchy	cauchy
Chi-Quadrat	chisq
Exponential	exp
F	f
Gamma	gamma
Geometrische	geom
Hypergeometrische	hyper
Logistische	logis
Lognormal	lnorm
Negative Binomial	nbinom
Normal	norm
Poisson	pois
Students t	t
Gleich	unif
Weibull	weibull

### Q Tiefer einsteigen

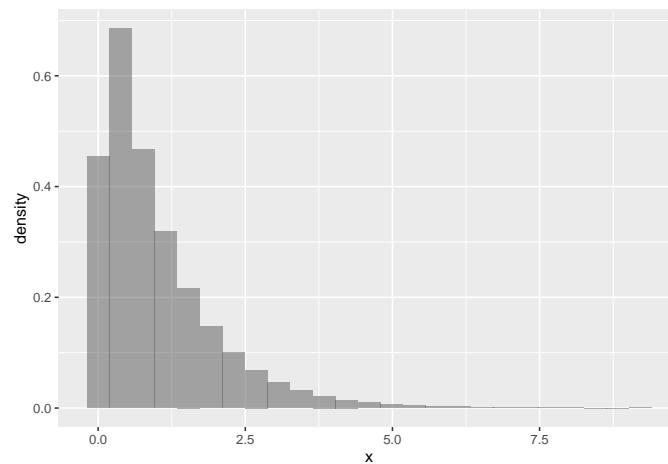
Die Funktion `gf_fitdistr()` erleichtert das Schätzen von Parametern für viele Verteilungen.

Die Funktion `gf_dist()` kann genutzt werden, um Verteilungen in unterschiedlichen Weisen darzustellen.

```
gf_dist('norm', mean = 100, sd = 10, kind = 'cdf')
```



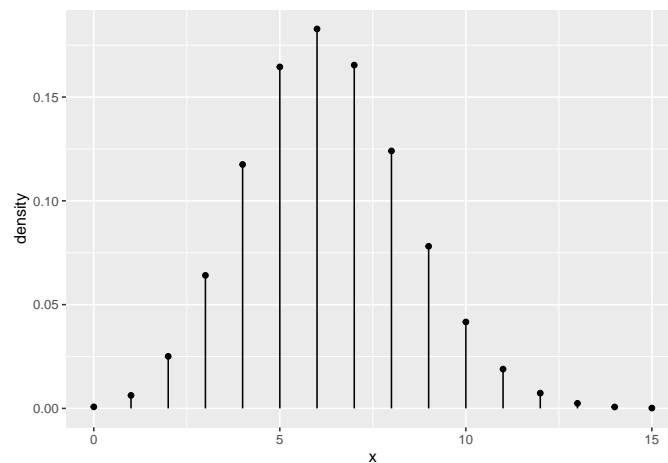
```
gf_dist('exp', kind = 'histogram', xlab = "x")
```



So wird der Parameter **rate** für die Skalierung der Verteilung standardmäßig auf 1 gesetzt und ist damit gleichbedeutend zu folgendem Befehl:

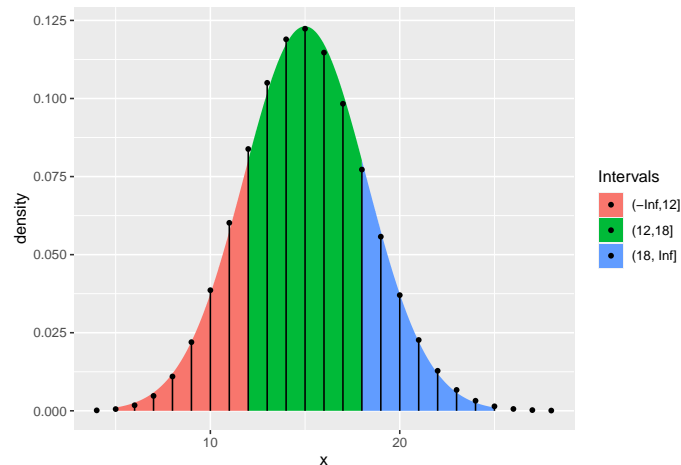
```
gf_dist('exp', rate = 1, kind = 'histogram', xlab = "x")
```

```
gf_dist('binom', size = 25, prob = 0.25, xlim = c(-1, 26))
```



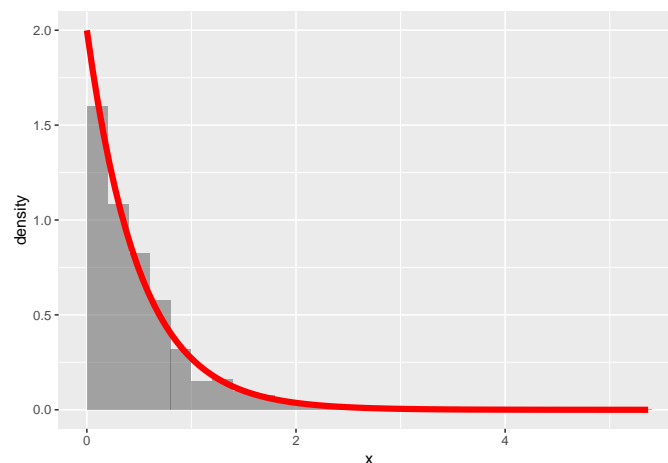
Mehrere Verteilungen können in einer Grafik angezeigt werden:

```
gf_dist("norm", mean = 50 * .3, sd = sqrt(50 * .3 * .7),
       fill = ~ cut(x, c(-Inf, 15 - 3, 15 + 3, Inf)), geom = "area") %>%
  gf_dist("binom", size = 50, prob = .3, col = "black", pch = 16) %>%
  gf_labs(fill = "Intervals")
```



Die Funktion `gf_fun()` kann genutzt werden, um eine beliebige Funktion darzustellen (hier im Beispiel eine exponentielle Zufallsvariable).

```
f <- makeFun(2 * exp(-2 * x) ~ x) # exponential with rate parameter 2
x <- rexp(1000, rate = 2)
gf_dhistogram(~ x, binwidth = 0.2, center = 0.1) %>%
  gf_fun(f(x) ~ x, color = "red", size = 2, xlim = c(0, 3))
```



## 12 Power-Berechnungen

Obwohl dieses Thema i. d. R. kein Hauptpunkt in einführenden Statistik-Kursen ist, helfen die Power-Berechnung (= Bestimmung der Trennschärfe eines Tests) und die Bestimmung des notwendigen Stichprobenumfangs, zentrale Konzepte in der Statistik zu vertiefen. In diesem Kapitel werden wir herausfinden, wie R genutzt werden kann, Power-Berechnungen über analytische Ansätze vorzunehmen. Als Beispiel nehmen wir ein einfaches Problem mit zwei Tests (t-Test und Vorzeichen-Test), die hier gerichtet bzw. einseitig angewendet werden. Wir werden die Power des Vorzeichen-Tests und die Power eines Tests basierend auf der Normalverteilungsannahme (einseitiger Einstichproben-t-Test) vergleichen unter der Annahme, dass die Varianz  $\sigma$  der Population bekannt sei.  $X_1, \dots, X_{25}$  seien u.i.v.  $N(0.3, 1)$  (das ist die Zufallsfolge, für die wir die Power berechnen wollen). Wir testen die Nullhypothese  $H_0 : \mu = 0$  versus  $H_A : \mu > 0$  zum Signifikanzniveau  $\alpha = 0.05$ .

### 12.1 Vorzeichen-Test

Wir starten mit der Berechnung des Alpha-Fehlers (Fehler 1. Art) für den Vorzeichen-Test. Wir verwerfen die Nullhypothese, wenn die Anzahl der positiven Werte groß ist. Unter der Nullhypothese ist das eine binomialverteilte Zufallsvariable mit  $n = 25$  Versuchen und  $p = 0.5$  Wahrscheinlichkeit für einen positiven Wert. Lassen Sie uns Werte zwischen 15 und 19 nutzen.

```
xvals <- 15:19
probs <- 1 - pbinom(xvals, size = 25, prob = 0.5)
cbind(xvals, probs)
```

	xvals	probs
[1,]	15	0.114761
[2,]	16	0.053876
[3,]	17	0.021643
[4,]	18	0.007317
[5,]	19	0.002039

```
qbinom(.95, size = 25, prob = 0.5)
```

```
[1] 17
```

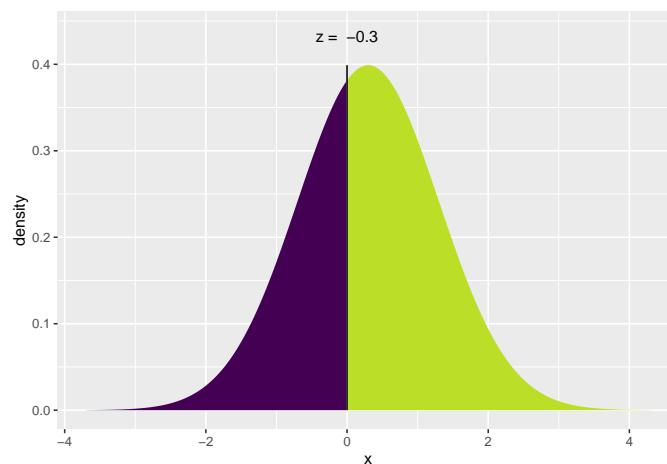
Wenn wir entscheiden die Nullhypothese zu verwerfen, sobald die Anzahl der positiven Werte 17 oder größer ist, sehen wir, dass wir ein  $\alpha$ -Niveau von 0.054 erreichen (Wert von `pbinom()` bei 16), der nahe am nominellen Wert der Fragestellung liegt. Wir berechnen die Power des Vorzeichen-Tests wie folgt: Die Wahrscheinlichkeit, dass  $X_i > 0$ , gegeben  $H_A$ , zutrifft, ergibt sich wie folgt:

```
1 - pnorm(0, mean = 0.3, sd = 1)
```

```
[1] 0.6179
```

Wir können das mit folgendem Befehl auch grafisch betrachten:

```
xpnorm(0, mean = 0.3, sd = 1, lower.tail = FALSE)
```



```
[1] 0.6179
```

Die Power unter der Alternativhypothese ist gleich der Wahrscheinlichkeit, 17 oder mehr positive Werte zu ziehen, gegeben  $p = 0.6179$ :

```
1 - pbinom(16, size = 25, prob = 0.6179)
```

```
[1] 0.3378
```

Die Power ist hier bestenfalls bescheiden.

## 12.2 t-Test

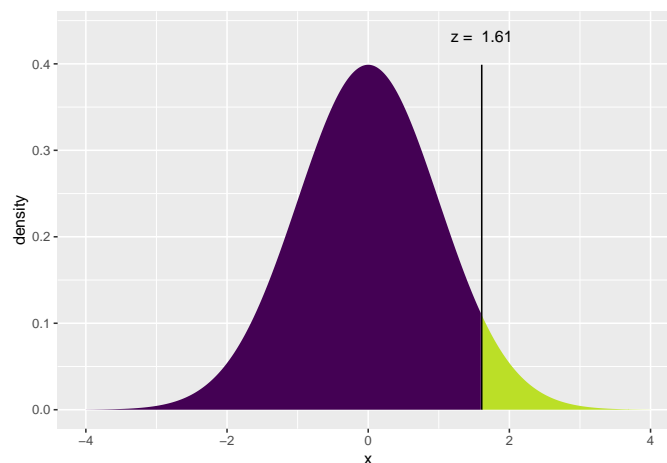
Als nächstes berechnen wir die Power für den Test, der auf der Normalverteilungsannahme basiert. Um den Vergleich fair zu gestalten, werden wir unser  $\alpha$ -Niveau auf 0.05388 setzen.

```
alpha <- 1 - pbinom(16, size = 25, prob = 0.5)
alpha
```

```
[1] 0.05388
```

Zunächst bestimmen wir den Ablehnungsbereich.

```
n <- 25
sigma <- 1 # given
stderr <- sigma/sqrt(n)
zstar <- xqnorm(1 - alpha, mean = 0, sd = 1)
```





```
zstar
```

```
[1] 1.608
```

```
crit <- zstar * stderr  
crit
```

```
[1] 0.3217
```

Daher verwerfen wir die Nullhypothese für beobachtete Mittelwerte größer als 0.322. Um die Power dieses einseitigen Tests zu berechnen, bestimmen wir die Wahrscheinlichkeit unter der Annahme der Alternativhypothese, rechts von dieser Obergrenze zu sein.

```
power <- 1 - pnorm(crit, mean = 0.3, sd = stderr)  
power
```

```
[1] 0.4568
```

Die Power des Tests basierend auf der Normalverteilungsannahme ist 0.457. Zur Überprüfung (oder für zukünftige ähnliche Berechnungen) können wir die Funktion `power.t.test()` nutzen.

```
power.t.test(n = 25, delta = .3, sd = 1, sig.level = alpha,  
             alternative = "one.sided", type = "one.sample")$power
```

```
[1] 0.4408
```

Der analytische (formelbasierte) Ansatz ergibt eine ähnliche Schätzung wie die direkte Berechnung über `power.t.test()`. Insgesamt sehen wir, dass der t-Test eine größere Power als der Vorzeichen-Test hat, wenn die zugrundeliegenden Daten tatsächlich normalverteilt sind.

# Die empirische Berechnung der Power zeigt die Stärke von Simulationen.

# 13 Datenmanagement

Datenmanagement ist eine Schlüsselkompetenz, die es Studierenden (und Dozierenden) ermöglicht “mit Daten zu arbeiten” oder wie Diane Lambert von Google sagte “mit Daten zu denken”. Wir neigen dazu, die Datenverwaltung für die Studierenden zu Beginn eines einführenden Statistikkurses auf ein Minimum zu beschränken und die Themen, wo nötig, schrittweise einzuführen. In Kursen, in denen die Studierenden substantielle und umfangreiche Projekte durchführen, wird ein größerer Fokus auf das Datenmanagement benötigt. In diesem Kapitel werden einige wichtige Möglichkeiten des Datenmanagements beschrieben.

## 13.1 Überprüfung von Dataframes

# Das Buch *Start Teaching with R* [5] enthält einen umfangreichen Abschnitt über die Datenverwaltung, einschließlich der Verwendung der Funktion `read.file()` zum Laden von Daten in R und RStudio.

### Q Tiefer einsteigen

Die Pakete `dplyr` und `tidyr` bieten einen eleganten Ansatz für das Datenmanagement und erleichtern den Studierenden die Möglichkeit, mit Daten zu arbeiten. Hadley Wickham, Autor der Pakete, schlägt vor, dass es sechs Schlüsselwörter (oder Verben) gibt, die in diesen Paketen implementiert sind, die es ermöglichen, eine große Anzahl von Aufgaben zu erfüllen: Filtern (`filter` - Zeilenabgleichskriterien beibehalten), Auswählen (`select` - Spalten nach Namen auswählen), Anordnen (`arrange` - Zeilen neu ordnen), Mutieren (`mutate` - neue Variablen hinzufügen), Zusammenfassen (`summarise` - Variablen auf Werte reduzieren) und Gruppieren nach (`group by` - Gruppen zusammenfassen). Siehe <http://www.amherst.edu/~nhorton/precursors> für weitere Details und Ressourcen.

Die Funktion `inspect()` kann bei der Beschreibung der Variablen in einem Dataframe (der Name für einen Datensatz in R) hilfreich sein.

```
inspect(iris)
```

categorical variables:

	name	class	levels	n	missing	distribution
1	Species	factor	3	150	0	1 setosa (33.3%), versicolor (33.3%) ...

quantitative variables:

	name	class	min	Q1	median	Q3	max	mean	sd	n	missing
...1	Sepal.Length	numeric	4.3	5.1	5.80	6.4	7.9	5.843	0.8281	150	0
...2	Sepal.Width	numeric	2.0	2.8	3.00	3.3	4.4	3.057	0.4359	150	0
...3	Petal.Length	numeric	1.0	1.6	4.35	5.1	6.9	3.758	1.7653	150	0
...4	Petal.Width	numeric	0.1	0.3	1.30	1.8	2.5	1.199	0.7622	150	0

Der iris-Datensatz enthält eine kategoriale und vier quantitative Variablen.

## 13.2 Hinzufügen neuer Variablen zu einem Dataframe

Wir können zusätzliche Variablen zu einem bestehenden Dataframe mit `mutate()` hinzufügen. Aber zuerst erstellen wir eine kleinere Version des iris-Datensatzes.

```
irisSmall <- select(iris, Species, Sepal.Length)
```

```
# schneidet die Daten in n Gruppen
irisSmall <- mutate(irisSmall, Length = cut(Sepal.Length, breaks=4:8))
```

Mehrere Befehle können mit dem `%>%` (PIPE-) Operator miteinander verknüpft werden:

```
irisSmall <- iris %>%
  select(Species, Sepal.Length) %>%
  mutate(Length = cut(Sepal.Length, breaks=4:8))
```

Beachten Sie, dass bei dieser Verwendung das erste Argument von `select()` die erste Variable ist (da sie die Daten von der vorherigen PIPE erbt).

```
head(irisSmall)
```

	Species	Sepal.Length	Length
1	setosa	5.1	(5,6]
2	setosa	4.9	(4,5]
3	setosa	4.7	(4,5]
4	setosa	4.6	(4,5]
5	setosa	5.0	(4,5]
6	setosa	5.4	(5,6]



### Hinweis

`cut()` ist eine Funktion, die verwendet werden kann, um aussagekräftigere Namen für die Gruppen zu definieren.

Der CPS85-Datensatz enthält Daten aus einer aktuellen Bevölkerungserhebung (aktuell also 1985). Zwei der Variablen in diesem Dataframe sind Alter (**age**) und Bildung (**educ**). Wir können die Anzahl der Jahre schätzen, die ein Arbeitnehmer im Erwerbsleben tätig ist, wenn wir davon ausgehen, dass er seit dem Abschluss seiner Ausbildung im Erwerbsleben tätig ist und dass sein Alter bei Abschluss der Ausbildung um 6 Jahre höher ist als die Anzahl der erreichten Ausbildungsjahre. Wir können dies als neue Variable in den Dataframe mit `mutate()` hinzufügen.

```
CPS85 <- mutate(CPS85, workforce.years = age - 6 - educ)
favstats(~ workforce.years, data=CPS85)
```

min	Q1	median	Q3	max	mean	sd	n	missing
-4	8	15	26	55	17.81	12.39	534	0

Tatsächlich wurde dies für alle bis auf einen Fall getan, um die bereits in den CPS85-Daten enthaltene Variable **exper** zu generieren.

```
tally(~ (exper - workforce.years), data = CPS85)
```

(exper - workforce.years)	
0	4
533	1

### 13.3 Variablen löschen

Da wir die Variable `exper` bereits haben, gibt es keinen Grund unsere neue Variable zu behalten. Eliminieren wir diese also wieder. Beachten Sie die geschickte Verwendung des Minuszeichens.

```
names(CPS85)
```

```
[1] "wage"      "educ"      "race"      "sex"
[5] "hispanic"  "south"     "married"   "exper"
[9] "union"     "age"       "sector"    "workforce.years"
```

```
CPS1 <- select(CPS85, select = -matches("workforce.years"))
names(CPS1)
```

```
[1] "wage"      "educ"      "race"      "sex"      "hispanic" "south"
[7] "married"   "exper"     "union"     "age"       "sector"
```

Beliebig viele Variablen können gelöscht oder auf ähnliche Weise beibehalten werden.

```
CPS1 <- select(CPS85, select = -matches("workforce.years|exper"))
```

### 13.4 Variablen umbenennen

Die Spaltennamen (Variablen) für einen Dataframe können mit der Funktion `rename()` im Paket `dplyr` geändert werden.

```
names(CPS85)
```

```
[1] "wage"      "educ"      "race"      "sex"
[5] "hispanic"  "south"     "married"   "exper"
[9] "union"     "age"       "sector"    "workforce.years"
```

```
CPSnew <- rename(CPS85, workforce = workforce.years)
names(CPSnew)
```

```
[1] "wage"      "educ"      "race"      "sex"      "hispanic"  "south"
[7] "married"   "exper"     "union"     "age"      "sector"    "workforce"
```

Die Zeilenamen eines Dataframes können durch einfache Zuweisung mittels `row.names()` geändert werden.

Der `faithful`-Datensatz (im Paket `datasets`, welches immer verfügbar ist) hat sehr unglückliche Namen.

```
names(faithful)
```

```
[1] "eruptions" "waiting"
```

#### **Tiefer einsteigen**

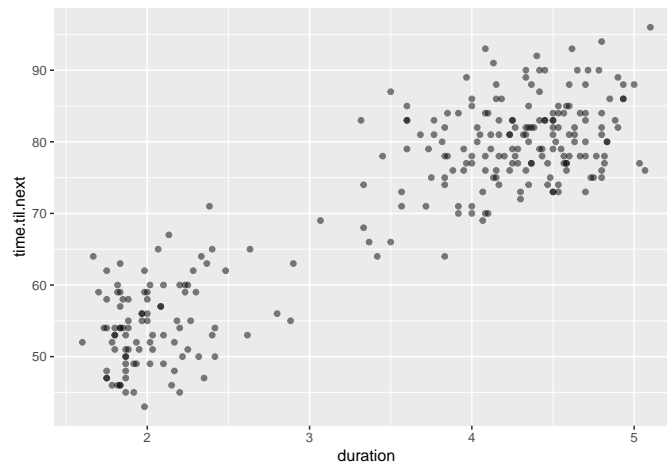
Es ist ratsam, vom ersten Tag an Praktiken für die Auswahl von Variablennamen festzulegen.

Die Messungen sind die Dauer einer Eruption und die Zeit bis zur folgenden Eruption, also vergeben wir geeignetere Namen.

```
faithful <- rename(faithful,
                   duration = eruptions,
                   time.til.next = waiting)
names(faithful)
```

```
[1] "duration"      "time.til.next"
```

```
library(mosaic)
gf_point(time.til.next ~ duration, alpha = 0.5, data = faithful)
```



Wenn eine Variable, die einen Datensatz enthält, modifiziert oder verwendet wird, um ein anderes Objekt zu speichern, können die Originaldaten aus dem entsprechenden Paket mittels `data()` wiederhergestellt werden.

```
data(faithful)
head(faithful, 3)
```

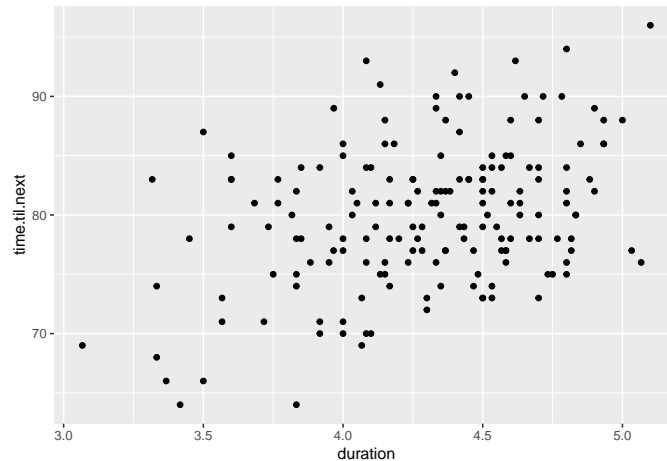
	eruptions	waiting
1	3.600	79
2	1.800	54
3	3.333	74

## 13.5 Erstellen von Teilmengen spezifischer Beobachtungen

Wir können auch `filter()` verwenden, um die Größe eines Dataframes zu reduzieren, indem nur bestimmte Zeilen ausgewählt werden.

```
data(faithful)
names(faithful) <- c('duration', 'time.til.next')

# zur Erstellung von Teilmengen kann jede Logik verwendet werden
faithfulLong <- filter(faithful, duration > 3)
gf_point( time.til.next ~ duration, data = faithfulLong)
```



## 13.6 Sortieren von Dataframes

Dataframes können mit der Funktion `arrange()` sortiert werden.

```
head(faithful, 3)
```

	duration	time.til.next
1	3.600	79
2	1.800	54
3	3.333	74

```
sorted <- arrange(faithful, duration)
head(sorted, 3)
```

	duration	time.til.next
1	1.600	52
2	1.667	64
3	1.700	59



### Achtung!

Es ist normalerweise besser, neue Datensätze zu erstellen, als das Original zu modifizieren.



## 13.7 Zusammenfügen von Dataframes

Der `fusion1`-Datensatz im Paket `fastR` enthält Genotyp-Informationen für einen SNP (single nucleotide polymorphism) im Gen `TCF7L2`. Der Dataframe enthält Phänotypen (einschließlich Typ-2-Diabetes Fall-/Kontrollstatus) für eine sich überschneidende Menge von Individuen. Wir können diese zusammenfügen (oder verschmelzen), um die Assoziation zwischen Genotypen und Phänotypen mit Hilfe von `merge()` zu untersuchen.

```
library(fastR2)
fusion1 <- arrange(FUSION1, id)
head(fusion1, 3)
```

	id	marker	markerID	allele1	allele2	genotype	Adose	Cdose	Gdose	Tdose
1	1002	RS12255372	1	3	3	GG	0	0	2	0
2	1009	RS12255372	1	3	3	GG	0	0	2	0
3	1012	RS12255372	1	3	3	GG	0	0	2	0

```
data(pheno)
head(Pheno, 3)
```

	id	t2d	bmi	sex	age	smoker	chol	waist	weight	height	whr	sbp	dbp
1	1002	case	32.86	F	70.76	former	4.57	112.0	85.6	161.4	0.9868	135	77
2	1009	case	27.39	F	53.92	never	7.32	93.5	77.4	168.1	0.9397	158	88
3	1012	control	30.47	M	53.86	former	5.02	104.0	94.6	176.2	0.9327	143	89

```
library(tidyr)
fusion1m <- inner_join(fusion1, Pheno, by='id')
head(fusion1m, 3)
```

	id	marker	markerID	allele1	allele2	genotype	Adose	Cdose	Gdose	Tdose
1	1002	RS12255372	1	3	3	GG	0	0	2	0
2	1009	RS12255372	1	3	3	GG	0	0	2	0
3	1012	RS12255372	1	3	3	GG	0	0	2	0

	t2d	bmi	sex	age	smoker	chol	waist	weight	height	whr	sbp	dbp
1	case	32.86	F	70.76	former	4.57	112.0	85.6	161.4	0.9868	135	77
2	case	27.39	F	53.92	never	7.32	93.5	77.4	168.1	0.9397	158	88
3	control	30.47	M	53.86	former	5.02	104.0	94.6	176.2	0.9327	143	89

Jetzt sind wir bereit, mit der Analyse zu beginnen.

```
tally(~t2d + genotype, data = fusion1m)
```

	genotype		
t2d	GG	GT	TT
case	737	375	48
control	835	309	27

## 13.8 Extrahieren und Zusammenfassen von Informationen

Das Paket `tidyr` bietet eine flexible Möglichkeit, die Anordnung der Daten zu ändern. Es wurde für die Konvertierung zwischen langen und breiten Versionen von Zeitreihendaten entwickelt und seine Argumente sind in diesem Sinne benannt.

Eine häufige Situation ist, wenn wir von einer breiten Form in eine lange Form konvertieren wollen, weil wir die Perspektive darüber, was eine Beobachtungseinheit ist, ändern wollen. Zum Beispiel ist im `traffic`-Dataframe jede Zeile ein Jahr, und es werden Daten für mehrere Staaten bereitgestellt.

# Die Vignetten, die in den Paketen `tidyr` und `dplyr` enthalten sind, bieten eine Reihe nützlicher Beispiele für gängige Datenmanipulationen.

```
data(traffic)
Traffic
```

	year	cn.deaths	ny	cn	ma	ri
1	1951	265	13.9	13.0	10.2	8.0
2	1952	230	13.8	10.8	10.0	8.5
3	1953	275	14.4	12.8	11.0	8.5
4	1954	240	13.0	10.8	10.5	7.5
5	1955	325	13.5	14.0	11.8	10.0
6	1956	280	13.4	12.1	11.0	8.2
7	1957	273	13.3	11.9	10.2	9.4
8	1958	248	13.0	10.1	11.8	8.6
9	1959	245	12.9	10.0	11.0	9.0

Wir können dies so umformen, dass jede Zeile eine Messung für einen einzelnen Staat in einem Jahr enthält.

```
longTraffic <- Traffic %>%
  gather(state, deathRate, ny:ri)
head(longTraffic)
```

	year	cn.deaths	state	deathRate
1	1951	265	ny	13.9
2	1952	230	ny	13.8
3	1953	275	ny	14.4
4	1954	240	ny	13.0
5	1955	325	ny	13.5
6	1956	280	ny	13.4

Wir können auch in die andere Reichung umformen, diesmal mit allen Daten für einen bestimmten Staat in einer Zeile im Dataframe.

```
stateTraffic <- longTraffic %>%
  select(year, deathRate, state) %>%
  mutate(year=paste("deathRate.", year, sep="")) %>%
  spread(year, deathRate)
stateTraffic
```

	state	deathRate.1951	deathRate.1952	deathRate.1953	deathRate.1954
1	cn	13.0	10.8	12.8	10.8
2	ma	10.2	10.0	11.0	10.5
3	ny	13.9	13.8	14.4	13.0
4	ri	8.0	8.5	8.5	7.5

	deathRate.1955	deathRate.1956	deathRate.1957	deathRate.1958	deathRate.1959
1	14.0	12.1	11.9	10.1	10.0
2	11.8	11.0	10.2	11.8	11.0
3	13.5	13.4	13.3	13.0	12.9
4	10.0	8.2	9.4	8.6	9.0

## 13.9 Neue Variablen hinzufügen

Eine Reihe von Funktionen erleichtern die Erstellung oder Neucodierung von Variablen.

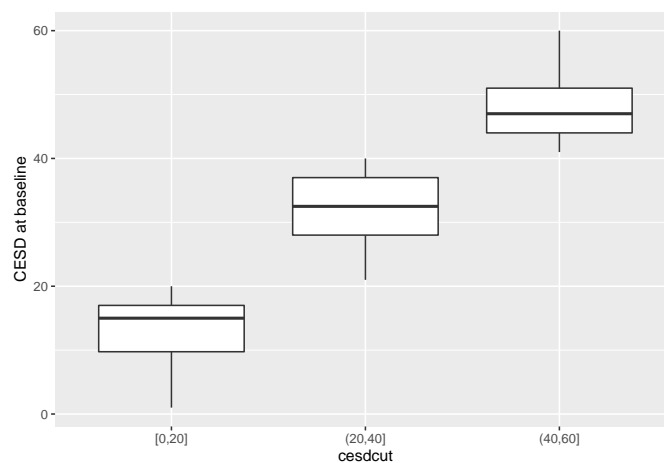
### 13.9.1 Kategoriale aus einer quantitativen Variable erstellen

Als nächstes zeigen wir, wie man eine dreistufige kategoriale Variable mit Trennung bei 20 und 40 für die CESD-Skala (die von 0 bis 60 Punkten reicht) erstellt.

```
favstats(~ cesd, data=HELPrct)
```

```
min Q1 median Q3 max mean sd n missing
1 25 34 41 60 32.85 12.51 453 0
```

```
HELPrct <- mutate(HELPrct,
  cesdcut = cut(cesd,
    breaks=c(0, 20, 40, 60),
    include.lowest=TRUE))
gf_boxplot(cesd ~ cesdcut, data=HELPrct)
```



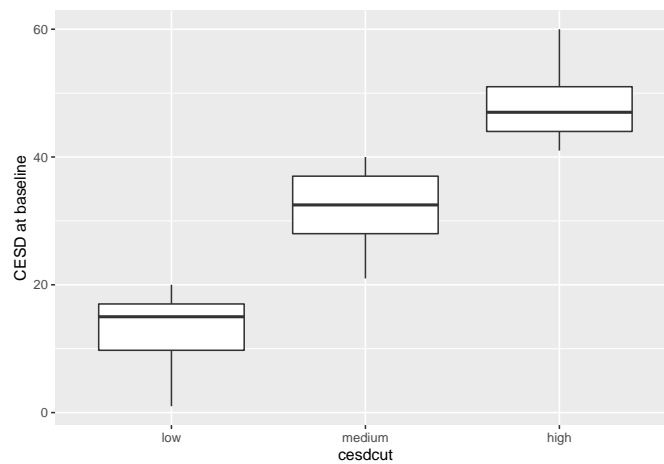
Es bietet sich an, die Labels besser zu bezeichnen.



#### Hinweis

Die Funktion `ntiles()` kann verwendet werden, um die Erstellung von Gruppen auf diese Weise zu automatisieren.

```
HELPrct <- mutate(HELPrct,
  cesdcut = cut(cesd,
    labels=c("low", "medium", "high"),
    breaks=c(0, 20, 40, 60),
    include.lowest=TRUE))
gf_boxplot(cesd ~ cesdcut, pch="|", data=HELPrct)
```



Die `derivedFactor()` Funktion ist noch allgemeiner und kann auch zu diesem Zweck verwendet werden.

```
HELPrct <- mutate(HELPrct,
  anothercut = derivedFactor(
    low = cesd >= 0 & cesd <= 20,
    medium = cesd > 20 & cesd <= 40,
    high = cesd > 40))
```

### 13.9.2 Faktoren neu ordnen

Standardmäßig verwendet R die erste Ebene in lexikographischer Reihenfolge als Referenzgruppe für die Modellierung. Dies kann mit der `relevel()` Funktion überschrieben werden (siehe auch `reorder()`).

```
tally(~ substance, data = HELPrct)
```

```
substance
alcohol cocaine heroin
      177      152      124
```

```
coef(lm(cesd ~ substance, data = HELPrct))
```

```
(Intercept) substancecocaine substanceheroin
      34.3729      -4.9518      0.4981
```

```
HELPrct <- mutate(HELPrct, subnew = relevel(substance, ref="heroin"))
coef(lm(cesd ~ subnew, data=HELPrct))
```

```
(Intercept) subnewalcohol subnewcocaine
      34.8710      -0.4981      -5.4499
```

## 13.10 Gruppenstatistiken

Es kann oft sinnvoll sein, zusammenfassende Statistiken nach Gruppen zu berechnen und diese in ein Dataframe einzufügen. Die Funktion `group_by()` im Paket `dplyr` erleichtert diesen Prozess. Hier zeigen wir, wie man eine Variable hinzufügt, die das mittlere Alter der Probanden je Drogenmittel enthält.

```
favstats(age ~ substance, data = HELPrct)
```

```
substance min Q1 median    Q3 max  mean    sd  n missing
1 alcohol  20 33   38.0 43.00  58 38.20 7.652 177      0
2 cocaine  23 30   33.5 37.25  60 34.49 6.693 152      0
3 heroin   19 27   33.0 39.00  55 33.44 7.986 124      0
```

```
ageGroup <- HELPrct %>%
  group_by(substance) %>%
  summarise(agebygroup = mean(age))
ageGroup
```

```
# A tibble: 3 x 2
  substance agebygroup
  <fct>      <dbl>
1 alcohol    38.2
2 cocaine    34.5
3 heroin     33.4
```

```
nrow(ageGroup)
```

```
[1] 3
```

```
nrow(HELPrct)
```

```
[1] 453
```

```
HELPmerged <- left_join(ageGroup, HELPrct, by = "substance")
favstats(agebygroup ~ substance, data = HELPmerged)
```

	substance	min	Q1	median	Q3	max	mean	sd	n	missing
1	alcohol	38.20	38.20	38.20	38.20	38.20	38.20	0	177	0
2	cocaine	34.49	34.49	34.49	34.49	34.49	34.49	0	152	0
3	heroin	33.44	33.44	33.44	33.44	33.44	33.44	0	124	0

```
nrow(HELPmerged)
```

```
[1] 453
```

## 13.11 Umgang mit fehlenden Werten

Fehlende Werte entstehen in fast allen Untersuchungen der realen Welt. R verwendet das NA-Zeichen als Indikator für fehlende Werte. Der `HELPmiss`-Datensatz innerhalb des Paketes `mosaicData` umfasst alle  $n = 470$  Probanden, die zu Studienbeginn eingeschrieben sind (einschließlich der  $n = 17$  Probanden mit einigen fehlenden Werten, die nicht in `HELPrct` enthalten waren).

```
smaller <- select(HELPmiss, cesd, drugrisk, indtot, mcs, pcs, substance)
dim(smaller)
```

```
[1] 470    6
```

```
summary(smaller)
```

cesd		drugrisk		indtot		mcs		pcs	
Min.	: 1.0	Min.	: 0.00	Min.	: 4.0	Min.	: 6.76	Min.	:14.1
1st Qu.:	25.0	1st Qu.:	0.00	1st Qu.:	32.0	1st Qu.:	21.66	1st Qu.:	40.4
Median	:34.0	Median	: 0.00	Median	:37.5	Median	:28.56	Median	:48.9
Mean	:32.9	Mean	: 1.87	Mean	:35.7	Mean	:31.55	Mean	:48.1
3rd Qu.:	41.0	3rd Qu.:	1.00	3rd Qu.:	41.0	3rd Qu.:	40.64	3rd Qu.:	57.0
Max.	:60.0	Max.	:21.00	Max.	:45.0	Max.	:62.18	Max.	:74.8
		NA's	:2	NA's	:14	NA's	:2	NA's	:2

substance  
 alcohol:185  
 cocaine:156  
 heroin :128  
 missing: 1

Von den 470 Probanden in den 6 Variablen des Dataframes haben nur die Variablen `drugrisk`, `indtot`, `mcs`, und `pcs` fehlende Werte

```
favstats(~ mcs, data = smaller)
```

min	Q1	median	Q3	max	mean	sd	n	missing
6.763	21.66	28.56	40.64	62.18	31.55	12.78	468	2

```
with(smaller, sum(is.na(mcs)))
```

```
[1] 2
```



```
nomiss <- na.omit(smaller)
dim(nomiss)
```

```
[1] 453  6
```

```
nrow(nomiss)
```

```
[1] 453
```

```
ncol(nomiss)
```

```
[1] 6
```

```
favstats(~ mcs, data = nomiss)
```

	min	Q1	median	Q3	max	mean	sd	n	missing
	6.763	21.79	28.6	40.94	62.18	31.7	12.82	453	0

Alternativ könnten wir den gleichen Datenbestand mit Hilfe von logischen Bedingungen erzeugen.

```
nomiss <- filter(smaller,
                  (!is.na(mcs) & !is.na(indtot) & !is.na(drugrisk)))
dim(nomiss)
```

```
[1] 453  6
```

## 14 Fallstudie Gesundheitsevaluation (HELP-Studie)

Viele Beispiele in diesem Buch greifen auf Daten der HELP-Studie zurück. Bei dieser Studie handelt es sich um eine randomisierte klinische Studie mit Patienten aus einer stationären Drogenentzugsanstalt. Patienten ohne laufende allgemeinmedizinische Versorgung wurden entweder einer Gruppe zugelost, in der eine multidisziplinäre Untersuchung und eine kurze Motivationsintervention vorgesehen war, oder einer Gruppe mit Standardbehandlung.<sup>1</sup> Die Ergebnisse der Studie wurden zusammen mit einer Reihe von zusätzlichen Analysen veröffentlicht [9, 7, 6].

Teilnehmer an der Studie mussten erwachsen sein, Spanisch oder Englisch sprechen und Alkohol, Heroin oder Kokain als ihre Hauptdroge angeben. Außerdem mussten sie in der Nähe in der Klinik wohnen, in der sie behandelt werden sollten, oder obdachlos sein. Selektionskriterien waren: a) laufende Versorgung und Absicht, diese weiterzuführen, b) fortgeschrittene Demenz, c) konkrete Pläne, die Gegend um Boston zu verlassen, so dass sie nicht an einer Behandlung teilnehmen konnten, d) keine Angabe von Kontaktinformationen oder e) Schwangerschaft.

Die Teilnehmer wurden bei der Aufnahme zum Entzugsaufenthalt befragt; alle sechs Monate für die Dauer von zwei Jahren wurden die Teilnehmer erneut befragt. Dabei wurde eine Anzahl von metrisch-, ordinal- und nominal-skalierten sowie Daten für Überlebenszeitanalysen<sup>2</sup> zu jedem der fünf Zeitpunkte erhoben. Die Ethikkommission des Medizinischen Zentrums der Universität Boston überprüfte das Studienkonzept, insbesondere die Erstellung eines anonymisierten Datensatzes. Das Zertifikat für Datenschutz, ausgestellt durch das *Department of Health and Human Services*, bescheinigt weitere Datenschutzmaßnahmen.

Das R-Paket `mosaicData` enthält mehrere Varianten des anonymisierten HELP-Datensatzes. Für die vorliegende Fallstudie konzentrieren wir uns auf den `HELPprct`-Datensatz, der 27 Variablen zu 453 Personen enthält, wobei vergleichsweise wenig Daten fehlen, besonders zum ersten Messzeitpunkt. Tabelle 14.1. Weitere Informationen finden sich online unter <http://www.amherst.edu/~nhorton/r2>. Die Messinstrumente sind hier aufgeführt: <http://www.amherst.edu/~nhorton/help>.

---

<sup>1</sup>Die HELP-Studie wurde finanziert durch das *National Institute on Alcohol Abuse and Alcoholism* (R01-AA10870, Samet PI) sowie das *National Institute on Drug Abuse* (R01-DA10019, Samet PI).

<sup>2</sup>Survival Time Analysis

Tabelle 14.1: Variablen des Datensatzes zur HELP-Studie

Variable	Beschreibung	Hinweis
1 'alter'	Alter zu Messzeitpunkt 1 (in Jahren) (Spannweite 19-60)	
2 'anysub'	Einnahme von Drogen nach Beginn der Intervention	vgl. 'daysanysub'
3 'cesd'	Depressionswert (höhere Werte zeigen schwerere depressive Symptome an) (Spannweite 0-60)	
4 'd1'	Anzahl der Einlieferungen aufgrund medizinischer Probleme (Spannweite 0-100)	
5 'daysanysub'	Zeit (in Tagen) bis zur ersten Einnahme von Drogen nach Beginn der Intervention	vgl. 'anysubstatus'
6 'dayslink'	Zeit (in Tagen) bis zur Vermittlung einer allgemeinmedizinischer Versorgung (Spannweite 0-456)	vgl. 'linkstatus'
7 'drugrisk'	Risiko-Assessment-Testbatterie (RAB) zum Drogenrisiko (Spannweite 0-21)	vgl. 'sexrisk'
8 'e2b'	Anzahl der Teilnahmen an einem Drogenentzugsprogramm in den letzten sechs Monaten (Spannweite 0-21)	
9 'female'	Geschlecht des Teilnehmers (0: Mann - 1: Frau)	
10 'g1b'	Vorhandensein von Suizidgedanken (in den letzten 30 Tagen) (Werte 0: nein - 1: ja)	
11 'homeless'	Eine oder mehr Nächte auf der Straße oder in einer Notunterkunft in den letzten sechs Monaten (0: nein - 1: ja)	
12 'i1'	Mittlere Anzahl alkoholischer Getränke (in Standardeinheiten) pro Tag (in den letzten 30 Tagen) (Spannweite 0-142)	
13 'i2'	Maximale Anzahl alkoholischer Getränke (in Standardeinheiten) pro Tag (in den letzten 30 Tagen) (Spannweite 0-184)	
14 'id'	zufällige Personennummer (Spannweite 1-470)	
15 'indtot'	Gesamtwert für das Drogenfolgeninventar (Inventory of Drug Use Consequences (InDUC) total score) (Spannweite 4-45)	
16 'linkstatus'	Erfolgreiche Vermittlung einer allgemeinmedizinischen Versorgung nach der Entzugsintervention (0: nein - 1: ja)	vgl. 'dayslink'
17 'mcs', Wert für die Skala 'Psychisches Wohlbefinden' (Mental Component Score) des Short Form Gesundheitsfragebogen (SF-36) (Spannweite 7-62), vgl. 'pcs';		
18 'pcs', Wert für die Skala 'Körperliche Funktionsfähigkeit' (Physical Component Score) des Short Form Gesundheitsfragebogen (SF-36) (Spannweite 7-62), vgl. 'mcs';		
19 'ps_fr'	Wahrgenommener sozialer Rückhalt (durch Freunde) (Spannweite 0-14)	
20 'racegrp'	Rasse/Ethnie (Schwarz/Weiß/Latino/Sonstige)	
21 'satreat'	laufende Behandlung durch staatliches Hilfsprogramm für Drogenabhängige zu Beginn der Studie (0: nein - 1: ja)	
22 'sex'	Geschlecht (male: männlich - female: weiblich)	
23 'sexrisks'	Wert zum Risiko sexuell übertragbarer Krankheiten (Risk-Assessment Battery (RAB) sex risk score) (Spannweite 0-21)	
24 'substance'	Primär genutzte Droge (Alkohol/Kokain/Heroin)	
25 'treat'	Interventionsgruppe in der Studie (Experimentalgruppe	d.h. HELP-Klinik: ja - Standardbehandlung: no)
27		Die (empirische) Spannweite ist für metrische Variablen angegeben (zum Zeitpunkt des Studienbeginns).;

Note:

# 15 Aufgaben

Die *erste* Nummer zu Beginn einer Aufgabe bezieht sich auf das Kapitel, zu der die Aufgabe gehört.

**3.1** Erstellen Sie ein facettiertes Histogramm der Depressivitätswerte (CESD-Scores), wobei die Drogenart die zu gruppierende Variable (d. h. die facettierte Variable) darstellt. Beschränken Sie Analyse auf die Männer und überlagern Sie das Histogramm mit einer Dichtekurve der Normalverteilung. Der Datensatz heißt `HELPrct`.

**5.1** Berechnen Sie eine einfache lineare Regression, um die Anzahl alkoholischer Getränke als Funktion des psychischen Wohlbefindens vorherzusagen (Datensatz `HELPrct`). Das Modell kann mit dieser Formelschreibweise spezifiziert werden: `i1 ~ mcs`. Untersuchen Sie die Verteilung der Residuen für dieses Modell.

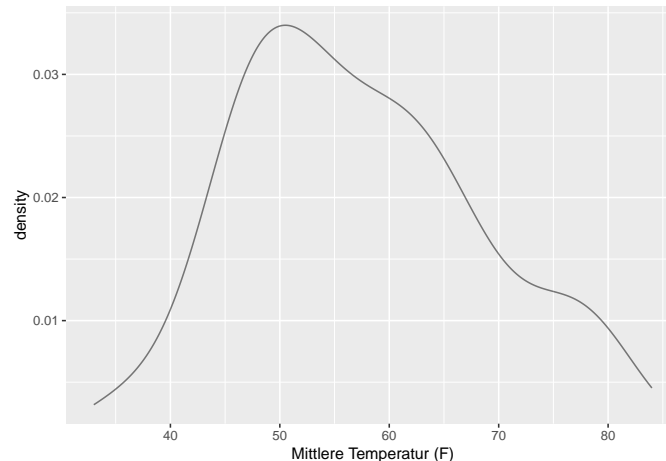
**10.1** Der Datensatz `RailTrail` (Teil des Paketes `mosaic`) beinhaltet eine Häufigkeitsauswertung zum Schienenverkehr für eine Trasse in Northampton, Massachusetts, während eines Zeitraums von 90 Tagen in 2005. Die Stadtverwaltung möchte die Auslastung des Schienennetzes wissen und verstehen, wie sie sich als Funktion von Temperatur und Wochentag ändert. Beschreiben Sie die Verteilung der Variablen `avgtemp` hinsichtlich ihres Mittelwerts, ihrer Streuung und Form.

```
data("RailTrail")
```

```
favstats(~ avgtemp, data = RailTrail)
```

```
min      Q1 median   Q3 max  mean    sd  n missing
33 48.62  55.25 64.5  84 57.43 11.33 90      0
```

```
gf_dens(~ avgtemp, xlab = "Mittlere Temperatur (F)", data = RailTrail)
```



10.2 Der Datensatz **RailTrail** beinhaltet eine Variable namens **cloudcover**, die den Grad der Bewölkung wiedergibt. Beschreiben Sie die Verteilung dieser Variable im Hinblick auf Lage, Streuung und Form.

10.3 Die Variable im Datensatz **RailTrail**, die die Höhe des Verkehrsaufkommens beschreibt, heißt **volume**. Beschreiben Sie die Lage, Streuung und Form dieser Variable.

10.4 Im Datensatz **RailTrail** gibt es eine Indikator-Variable, die anzeigt, ob ein Tag ein Wochentag war (**weekday == 1**) oder ein Wochenende bzw. Feiertag (**weekday == 0**). Setzen Sie **tally()** ein, um die Verteilung dieser kategorialen Variable zu beschreiben. Wie hoch ist der Anteil von Tagen des Wochenendes bzw. Feiertage?

10.5 Verwenden Sie gruppierte Boxplots, um die Verteilung von **volume** nach **weekday** zu untersuchen (Datensatz **RailTrail**). Was lässt sich ablesen? <sup>1</sup>

10.6 Erstellen Sie ein Dichtediagramm, das für beide Arten von Wochentagen jeweils ein Dichtediagramm zeigt. Die beiden Diagramme sollen sich überlappen. Tipp: Nutzen Sie das Argument **fill**.

10.7 Erstellen Sie ein Streudiagramm für das Verkehrsaufkommen (**volume**) als Funktion der Temperatur (Datensatz **RailTrail**). Fügen Sie sowohl eine Regressionsgerade als auch einen geglätteten Mittelwert (Lowess-Kurve) hinzu. Was können Sie über den Zusammenhang schließen?

10.8 Berechnen Sie eine multiple Regression für **volume** als Funktion von **cloudcover**, **avgtemp**, **weekday** und die Interaktion von Wochentag mit Durchschnittstemperatur. Unterstützen die Daten die Interaktion auf einem Signifikanzniveau von  $\alpha = 0.05$ ?

10.9 Verwenden Sie **makeFun()**, um die Anzahl von Fahrten vorherzusagen an einem Wochentag, bei einer mittleren Temperatur von 60° Fahrenheit und ohne Wolken. Überprüfen Sie Ihre Berechnung anhand der Modellkoeffizienten:

<sup>1</sup>Hinweis: Sie müssen die numerische Variable **weekday** in eine Faktorvariable (**as.factor()**) umwandeln, oder nutzen Sie die Option **horizontal = FALSE**.

```
coef(fm)
```

(Intercept)	weekdayTRUE	avgtemp	cloudcover
378.834	-321.116	2.313	-17.198
weekdayTRUE:avgtemp			
4.727			

10.10 Nutzen Sie `makeFun()` und `gf_fun()`, um die Zahl der vorhergesagten Fahrten sowohl an Wochentagen als auch an Wochenenden bzw. Feiertagen mit einer Temperaturspanne von 30 bis 80 Grad Fahrenheit an einem wolkigen Tag (`cloudcover = 10`) zu visualisieren .

10.11 Erstellen Sie ein Histogramm der Residuen des Regressionsmodells aus der vorherigen Aufgabe, um die Normalverteilung zu überprüfen. Die Dichte einer Normalverteilung soll in der Abbildung über das Histogramm gelegt werden.

10.12 Erstellen Sie ein Streudiagramm mit dem gleichen Modell; dabei sollen die Residuen als Funktion der vorhergesagten Werte dargestellt werden. Beurteilen Sie auf dieser Basis die (Erfüllung der) Linearitätsannahme sowie die Annahme der Varianzgleichheit.

10.13 Erstellen Sie ein Streudiagramm mit dem gleichen Modell wie eben; dabei sollen die Residuen als Funktion der mittleren Temperatur dargestellt werden. Beurteilen Sie auf dieser Basis die (Erfüllung der) Linearitätsannahme sowie die Annahme der Varianzgleichheit.

11.1 Erzeugen Sie eine Stichprobe mit 1000 zufällig gezogenen Werten aus einer Exponentialverteilung, wobei der Parameter der Verteilung 2 betragen soll. Berechnen Sie den Mittelwert dieser 1000 Fälle.

11.2 Finden Sie den Median einer exponentialverteilten Zufallsvariablen  $X$  mit einem Parameterwert von 10.

12.1 Berechnen Sie die Power eins zweiseitigen t-Tests für unabhängige Stichproben, wobei beide Verteilungen approximativ normalverteilt sind mit der gleichen Standardabweichung. Der Unterschied zwischen den Mittelwerten liege bei einer halben Standardabweichungseinheit. Gehen Sie von 25 Fällen pro Gruppe aus und einem Alpha-Niveau von 0.0539.

12.2 Berechnen Sie die nötige Stichprobengröße  $n$  für eine Power von 90% bei einem t-Test für zwei unabhängige Gruppen, wobei der Unterschied in den Mittelwerten bei  $1/4$  einer Standardabweichungseinheit liege ( $\alpha = 0.005$ ).

13.1 Erstellen Sie ein Streudiagramm für den Datensatz `faithful`, wobei die Eruptionszeit als Funktion von der Zeit seit der letzten Eruption dargestellt werden soll.

13.2 Der Datensatz `fusion2` im R-Paket `fastR` beinhaltet Genotypen für ein bestimmtes Einzelnukleotid-Polymorphismus (SNP). Fügen Sie die Datensätze `fusion1`, `fusion2` und

**Pheno** zu einen Datensatz zusammen. Beachten Sie, dass **fusion1** und **fusion2** über die gleichen Variablenamen verfügen. Es ist eine (mögliche) sinnvolle Herangehensweise, die Variablenamen nach dem Zusammenfügen umzubenennen, so dass der resultierende Datensatz einfacher zu verwenden ist. Bereinigen Sie den Datensatz um Duplikate oder unnötige Spalten.

## 16 Hinweise

Folgende Pakete wurden verwendet: TBD.



## Literatur

- [1] JJ Allaire u. a. *rmarkdown: Dynamic Documents for R*. R package version 2.5.3. 2020. URL: <https://github.com/rstudio/rmarkdown>.
- [2] Ben Baumer u. a. „R Markdown: Integrating A Reproducible Analysis Tool into Introductory Statistics“. In: *Technol Innov Stat Educ* 8 (Feb. 2014).
- [3] Nicholas Horton, Randall Pruim und Daniel Kaplan. „Start Teaching with R and RStudio“. In: (Aug. 2014). DOI: [10.13140/2.1.4414.6567](https://doi.org/10.13140/2.1.4414.6567).
- [4] Nicholas J. Horton, Benjamin S. Baumer und Hadley Wickham. „Taking a Chance in the Classroom: Setting the Stage for Data Science: Integration of Data Management Skills in Introductory and Second Courses in Statistics“. In: *CHANCE* 28.2 (2015), S. 40–50. DOI: [10.1080/09332480.2015.1042739](https://doi.org/10.1080/09332480.2015.1042739). eprint: <https://doi.org/10.1080/09332480.2015.1042739>. URL: <https://doi.org/10.1080/09332480.2015.1042739>.
- [5] Daniel Kaplan, Nicholas Horton und Randall Pruim. „Start Modeling with R“. In: (Aug. 2014). DOI: [10.13140/2.1.1137.8566](https://doi.org/10.13140/2.1.1137.8566).
- [6] S. G. Kertesz u. a. „Slowing the Revolving Door: Stabilization Programs Reduce Homeless Persons’ Substance Use After Detoxification.“ In: *Journal of Substance Abuse Treatment* 24.3 (2003), S. 197–207. DOI: [10.1016/S0740-5472\(03\)00026-6](https://doi.org/10.1016/S0740-5472(03)00026-6).
- [7] J. Liebschutz u. a. „The Relationship Between Sexual and Physical Abuse and Substance Abuse Consequences.“ In: *Journal of Substance Abuse Treatment* 22.3 (2002), S. 121–128. DOI: [10.1016/S0740-5472\(02\)00220-9](https://doi.org/10.1016/S0740-5472(02)00220-9).
- [8] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2020. URL: <https://www.R-project.org/>.
- [9] J. H. Samet u. a. „Linking Alcohol and Drug Dependent Adults to Primary Medical Care: A Randomized Controlled Trial of a Multidisciplinary Health Intervention in a Detoxification Unit.“ In: *Addiction* 98.4 (2003), S. 509–516. DOI: [10.1046/j.1360-0443.2003.00328.x](https://doi.org/10.1046/j.1360-0443.2003.00328.x).
- [10] E. R. Tufte. *The Visual Display of Quantitative Information*. 2nd. Cheshire, CT: Graphics Press, 2001.
- [11] Yihui Xie. *Dynamic Documents with R and knitr*. 2nd. ISBN 978-1498716963. Boca Raton, Florida: Chapman und Hall/CRC, 2015. URL: <http://yihui.org/knitr/>.