# Anleitung zum Mathematischen Seminar

## 1 Einleitung

Im Rahmen des mathematischen Seminars schreiben Sie eine Seminararbeit zu einem mathematischen Thema. Dabei geht es in erster Linie darum, ein mathematisch/naturwissenschaftliches Argument zu erarbeiten und auszuformulieren. Sie arbeiten zu zweit an Ihrem Thema, Sie arbeiten aber auch innerhalb des grösseren Seminar-Projektes, welches als Ziel das Seminar-Buch hat.

Sie sind daher auch nicht ganz frei in der Wahl der Werkzeuge, die Sie zur Formatierung der Seminararbeit verwenden, und erst recht nicht in der Art und Weise, wie Ihr Beitrag Eingang ins Gesamtprojekt findet.

Die Wahl dieser Werkzeuge ist Ihnen bereits abgenommen worden. Sie ist weder willkürlich noch durch persönliche Vorlieben diktiert, sondern entspricht dem Standard in Mathematik und Physik und vielen technischen Wissenschaften. Viele Zeitschriften in diesen Gebieten erwarten Manuskripte typischerweise formatiert in LaTeX, und auch an der HSR ist seine Verwendung vor allem im Studiengang E das Werkzeug der Wahl für Studien- und Bachelor-Arbeiten.

Die Zusammenarbeit der vielen Autoren wird vereinfacht durch das Versionsverwaltungssystem GIT. Es wurde ursprünglich von Linus Torvalds für die Verwaltung des Source Code des Linux Kernel geschrieben. Es hat sich aber mehr oder weniger zu einem Standard entwickelt, moderne Projekte werden meistens mit damit verwaltet.

Die meisten Studierenden arbeiten im Mathematischen Seminar zum ersten Mal an einem grösseren Dokumentationsprojekt zusammen, und kennen daher diese Werkzeuge noch nicht. Die folgenden Abschnitte versuchen, den Einstieg zu vereinfachen. Sie beschränken sich auf das unbedingt Nötige. Es gibt genügend weiterführende Literatur, mit der Sie Ihre Kenntnisse vertiefen können, und die allwissende Müllhalde, das Internet, bietet zahllose Quellen zur Vertiefung. Von besonderem Interesse im Zusammenhang mit LaTeX `http://tex.stackexchange.com`.

## 2 Die Seminararbeit

In diesem Abschnitt werden ein paar Hilfen zur Gestaltung der Arbeit mit LaTeX zusammengestellt.

### 2.1 Text

Der Text ist natürlich der zentrale Teil Ihrer Arbeit, und auch der Teil, wo sie im Team am erkennbarsten zusammen arbeiten können müssen. Sie gehen aus von den bereits bereitgestellten

Files `main.tex` und `main.bib` (für das Literaturverzeichnis) im Unterverzeichnis, das Ihrer Arbeit zugewiesen ist.

Ihren Text geben Sie als eine Mischung von reinem Text und sogenannte Kontroll-Sequenzen oder Makros ein. Die Makros ermöglichen Sonderzeichen darzustellen, die im Textzeichensatz nicht zur Verfügung stehen. Bei der Verwendung von ASCII zum Beispiel benötigen bereits die Umlaute ä, ö und ü die Kontroll-Sequenzen `"a`, `"o` und `"u`. Im Allgemeinen beginnen Kontrollsequenzen jedoch mit \. Formeln im Text werden mit $ eingerahmt, herausgestellte Formeln, die auf einer eigenen Zeile stehen, sogenannte Displays, werden wie folgt gesetzt:

```
\[
\int_{-\infty}^\infty e^{-x^2/2}\,dx = \sqrt{2\pi}
\]
```

Der zugehörige Output ist

$$\int_{-\infty}^{\infty} e^{-x^2/2}\,dx = \sqrt{2\pi}$$

Es steht eine riesige Menge von Makros zur Darstellung so ziemlich aller mathematischen Formeln zur Verfügung. Sie können für Ihre Zwecke zum Beispiel im Skript abkupfern, oder die vielen Tutorials auf dem Internet konsultieren, oder mit Google nach Hilfen suchen, die meistens auf Stackexchange führen.

Hier noch ein paar Empfehlungen, die ihren Code lesbarer machen und die Kollaboration vereinfachen:

- Jede Formel auf eine eigne Zeile. Erfahrungsgemäss ist das Vorgehen bei Änderungen am Text anders als bei Änderungen an Formeln, daher ist es sinnvoll, diese bereits im Source-Code auseinander zu halten.

- Jeder Satz auf eine neue Zeile. Da Kollaborationswerkzeuge typischerweise zeilenweise arbeiten, können Änderungen, die zwei verschiedene Autoren am Text machen, nur dann wieder zusammengeführt werden, wenn die beiden Autoren verschiedene Zeilen geändert haben. Wenn jeder Satz auf einer neuen Zeile beginnt, hat das zur Folge, dass Änderungen zusammengeführt werden können, wenn sie verschiedene Sätze betreffen.

- Ein neuer Absatz wird durch eine Leerzeile signalisiert, nicht durch irgendwelche Makro-Aufrufe. Insbesondere dürfen Sie nicht durch \\ einen neuen Absatz einleiten. Dieses Makro erzeugt nur einen Zeilenwechsel, TEX betrachtet den nachfolgenden Text als Teil des selben Absatzes. Das ist zum Beispiel daran erkennbar, dass der üblich Einzug am Anfang eines neuen Absatz nicht erscheint.

- Wenn die Trennung von TEX nicht automatisch richtig gemacht wird, dann fügen sie "discretionary hyphens" hinzu, eigentlich nur Instruktionen an TEX, wie ein Wort getrennt werden muss. Sie dürfen keine festen Trennungsstriche einfügen, denn wenn der Text umformatiert werden muss (zum Beispiel wenn aus drucktechnischen Gründen das Seitenformat geändert werden muss), dann entstehen plötzlich nicht getrennte Wörter mitten in einer Zeile, die Bindestriche enthalten.

- Kümmern Sie sich bei der Eingabe Ihres Textes nicht um die Platzierung auf der Seite, dies ist die Aufgabe von LATEX. Und wenn ihnen das optische Resultat nicht gefällt, dann ist das nicht Ihr Problem, sondern das Problem des Herausgebers des Seminar-Buches, der für die herausgeberische Aufarbeitung zuständig ist.

## 2.2 Bilder

Ein Bild sagt mehr als tausend Worte. Streben Sie also an, Ihre Erkenntnisse möglichst aussagekräftigen Bildern zu illustrieren und verständlich zu machen. Leider liefert die Google-Bildersuche nicht immer wirklich gute Bilder, geben Sie sich also nicht damit zufrieden, suchen Sie Darstellungen, die Ihre Ideen am besten ausdrückt.

Dies kann auch bedeuten, dass Sie eine graphische Darstellung selbst erzeugen müssen. Eine breite Palette von Softwareprodukten steht für diesen Zweck zur Verfügung, doch zeigt die Erfahrung, dass alle Produkte eine gewisse Übung verlangen, bis gute Resultate entstehen. In jedem Fall müssen Sie für gute Resultate Zeit investieren.

Die Abbildungen des Skriptes sind zu einem bedeutenden Teil mit Hilfe von Metapost erzeugt. Metapost ist eine Graphik-Sprache, die aus dem Metafont-System abgeleitet ist, mit dem die zu TEX gehörigen Schriften erzeugt wurden. Als Bestandteil jeder einigermassen vollständigen TEX-Distribution ist es eine naheliegende Wahl, wenngleich vielleicht nicht die einfachste. Die Möglichkeiten sind jedoch in vielen Fällen völlig ausreichend. Sie können die Beispiele im Skript als Basis für Ihre eigenen Entwicklung nehmen. Sehr viel vollständiger aber auch komplexer ist Tikz.

Was auch immer Sie verwenden, streben Sie an, dass die erzeugten Graphiken Vektor-Graphiken sind, damit sie bei jeder beliebigen Auflösung optimal dargestellt werden. Bei Buchdruck werden sehr viel höhere Auflösungen verwendet als beim Laserdruck, Pixelgraphiken können beim Laserdruck akzeptabel aussehen, bei höheren Auflösungen werden die Defizite sichtbar. JPG Bilder sind also dann zweckmässig für photographische Bilder.

Zudem sollten Sie in Ihren Bildern Transparenz vermeiden, da gewisse Druckmaschinen damit Schwierigkeiten haben. Der PDF-Standard, der in der Druckvorstufe üblich ist, ist deutlich älter als was von LATEX erzeugt wird.

## 2.3 Literaturverzeichnis

Jede Seminararbeit hat ihr eigenes Literaturverzeichnis. Um die Formatierung der einzelnen Einträge brauchen Sie sich nicht zu kümmern, auch dies übernimmt TEX. Sie müssen aber die bibliographischen Angaben in strukturierte Form bereitstellen. Dies geschieht im File `main.bib` im Verzeichnis ihrer Seminararbeit. Nehmen Sie die Beispiele im File `references.bib` als Vorlagen, weitere Informationen können Sie auch auf Stackexchange finden.

## 2.4 Präsentation

Ihre Erkenntnisse sollen Sie auch in einer Präsentation ihren Kollegen vorstellen. In der Wahl Ihrer Präsentationsmittel sind Sie frei, doch das Sie bereits für Ihre Seminararbeit LATEX eingesetzt haben, liegt es einigermassen nahe, dasselbe Werkzeug auch für die Präsentation zu verwenden. Dies erlaubt Ihnen Graphiken und Formeln direkt wieder zu verwenden.

Meistens wird für Präsentationen das `beamer`-Package verwendet. Im Internet finden Sie viele Beispiele oder sogar Vorlagen für seinen Einsatz, die Sie für Ihre Zwecke adaptieren können.

# 3 Werkzeuge

Eine Implementation der nachstehend beschriebenen Produkte für Ihre bevorzugte Plattform finden Sie im Internet.

## 3.1 Texteingabe

Die Wahl eines Editors ist eine sehr persönliche Sache, sie hat viel mit den eigenen Arbeitsgewohnheiten zu tun. Im Rahmen der Einschränkung von TEX sind sie völlig frei in der Wahl ihres Texteingabe-Werkzeuges.

Word oder ein anderes Office-Produkt ist allerdings genauso wenig für die Eingabe von TEX-Code geeignet wie für die Erstellung von Programmcode. Es ist auch nur beschränkt der Versionierung und Kollaboration zugänglich, wie sie in diesem Projekt angestrebt wird. Sie müssen daher einen Editor verwenden, der für Programm-Code geeignet ist.

Im FS17 wird erstmals mit utf8-Text gearbeitet, so wie er in Java schon seit sehr langer Zeit üblich ist. Es ist daher darauf zu achten, dass utf8-Zeichen, insbesondere Umlaute und korrekt codiert und wiedergegeben werden. Ist dies in Ihrem Editor nicht möglich, müssen die traditionellen Zeichensequenzen zur Codierung von Umlauten und anderen Sonderzeichen verwendet werden, also `"a` für ä, `"o` für ö, `"u` für ü usw.

Ausser gewöhnlichen Text-Editoren stehen für TEX auch verschiedene GUI-Tools zur Verfügung, naturgemäss sind da die persönlichen Vorlieben aber noch wichtiger. Leider sind diese GUI-Tools nicht alle gleich nützlich. Insbesondere sind nicht alle gleich gut mit Versionierung und Kollaborations-Tools verträglich.

## 3.2 Unix und Makefiles

Einer Tradition folgend verwalte ich das Projekt auf Unix Systemen (Linux und/oder Mac OS X). Die Beschreibung der Befehle zur Versionsverwaltung zum Bau des Skriptes beziehen sich auf die Unix/Mac Umgebung. Sie sind aber in analoger Form auch für Windows-Benutzer zuständig, ich bitte die geübten Windows-User, die wichtigsten Unterschiede als Hilfestellung zu diesem Dokument beizutragen, genau zu diesem Zweck ist es Bestandteil des Github Projektes.

In der Unix-Umgebung ist naheliegend, die Verwaltung der Abhängigkeiten mit dem immer vorhandenen Werkzeug `make` vorzunehmen. Die Makefiles beschreiben, wie die verschiedenen Produkte des Projektes aus den Source-Files erzeugt werden müssen. Es genügt, im `skript`-Verzeichnis den Befehl `make` zu geben, worauf das Programm `make` das gesamte Skript neu baut. Das `make`-Kommando leitet aus den Makefiles ab, was alles gemacht werden muss, wenn einzelne Files geändert werden. Windows-User müssen hier einen Extraschritt selbst gehen, und möglicherweise einzelne Kommandos von Hand eingeben, um zu einem vollständigen Build zu kommen.

Alternativ kann man sich auch eine virtuelle Maschine mit einem Linux-Betriebssystem einrichten. Moderne Linux-Distributionen enthalten alle Werkzeuge, die für die Formatierung des Buches nötig sind, darin eingeschlossen die Erzeugung aller Bilder. Da nur einfach Textfiles editiert werden müssen, kann jeder beliebige Text-Editor dazu verwendet werden, was auch relativ leicht zu erlernen ist. Linux ist gerade auch im Bereich Embedded Systems weit verbreitet, so dass eine gewisse Vertrautheit damit sicher keine Negativqualifikation ist[1].

[1] Die Elektrotechnik-Abteilung der ETH verwendet fast ausschliesslich Linux.

## 3.3 GIT

Die TEX-Files des Seminar-Projektes werden als Github-Projekt geführt, Sie finden es unter `https://github.com/AndreasFMueller/Seminar17`. Die später abgebildeten Screenshots stammen vom letzten Jahr und verwenden daher das Projekt `SeminarDGL`. Es ist zwar jedermann möglich, auf die Files dieses Projektes zuzugreifen, aber eigene Beiträge zum Projekt können Sie so nicht leisten. Sie sollten daher als erstes ein eigenes Account auf `https://github.com` einrichten.

### 3.3.1 Verzeichnis für Ihre Arbeit

In der File-Hierarchie des Projektes steht Ihnen ein eigenes Verzeichnis für Ihre Beiträge zur Verfügung. Sie finden es unter `skript/thema`, wenn `thema` die Kurzbezeichnung Ihres Themas ist. In diesem Verzeichnis befindet sich bereits ein File `main.tex` mit einer Vorlage. Sie können sofort beginnen, dieses File zu erweitern, oder sie können von diesem File aus weitere Files einlesen (mit dem `\input` Befehl). Ebenfalls steht bereits ein noch leeres File `main.bib` für das Literaturverzeichnis bereit. Zu Ihrem Verzeichnis können Sie beliebige Files hinzufügen, und die darin vorhandenen Files jederzeit ändern. Änderungen an anderen Files dürfen Sie jedoch nicht vornehmen, und als Projektadministrator werde ich Pull-Requests (siehe weiter unten) abweisen, die Änderungen an Files anderer Projektteilnehmer vornehmen.

### 3.3.2 Forking

Bevor Sie Änderungen an den Files vornehmen können, müssen Sie einen Fork des Projektes erstellen. Suchen Sie dazu das Projekt `SeminarDGL` des Benutzers `AndreasFMueller`. In der rechten oberen Ecke des Webinterfaces finden Sie einen Knopf **Fork** (Abbildung 1). Klick darauf erzeugt eine Kopie des Projektes in ihrem eigenen Account (Abbildung 2). In diesem Fork fügen Sie ihre Seminararbeit hinzu und bearbeiten Sie, bis sie bereit ist, in das übergeordnete Projekt integriert zu werden.

### 3.3.3 Cloning

Bevor Sie an den Files Änderungen anbringen, brauchen Sie eine lokale Kopie des Projektes. Sie erreichen das durch *Clonen*. Der Befehl dazu ist

```
> git clone https://github.com/SeminarTeilnehmer/SeminarDGL.git
Cloning into 'SeminarDGL'...
remote: Counting objects: 236, done.
remote: Total 236 (delta 0), reused 0 (delta 0), pack-reused 236
Receiving objects: 100% (236/236), 253.56 KiB | 157.00 KiB/s, done.
Resolving deltas: 100% (143/143), done.
Checking connectivity... done.
```

### 3.3.4 Commit

Ihre Änderungen werden permanenter Bestandteil des Projektes durch den Commit. Damit die anderen Projektteilnehmer verstehen, was Ihre Änderungen bewirken, verlangt der Commit eine kurze Beschreibung derselben. Gewöhnen Sie sich daher an, oft zu committen, damit die Änderung mit einem kurzen Statement beschrieben werden kann.

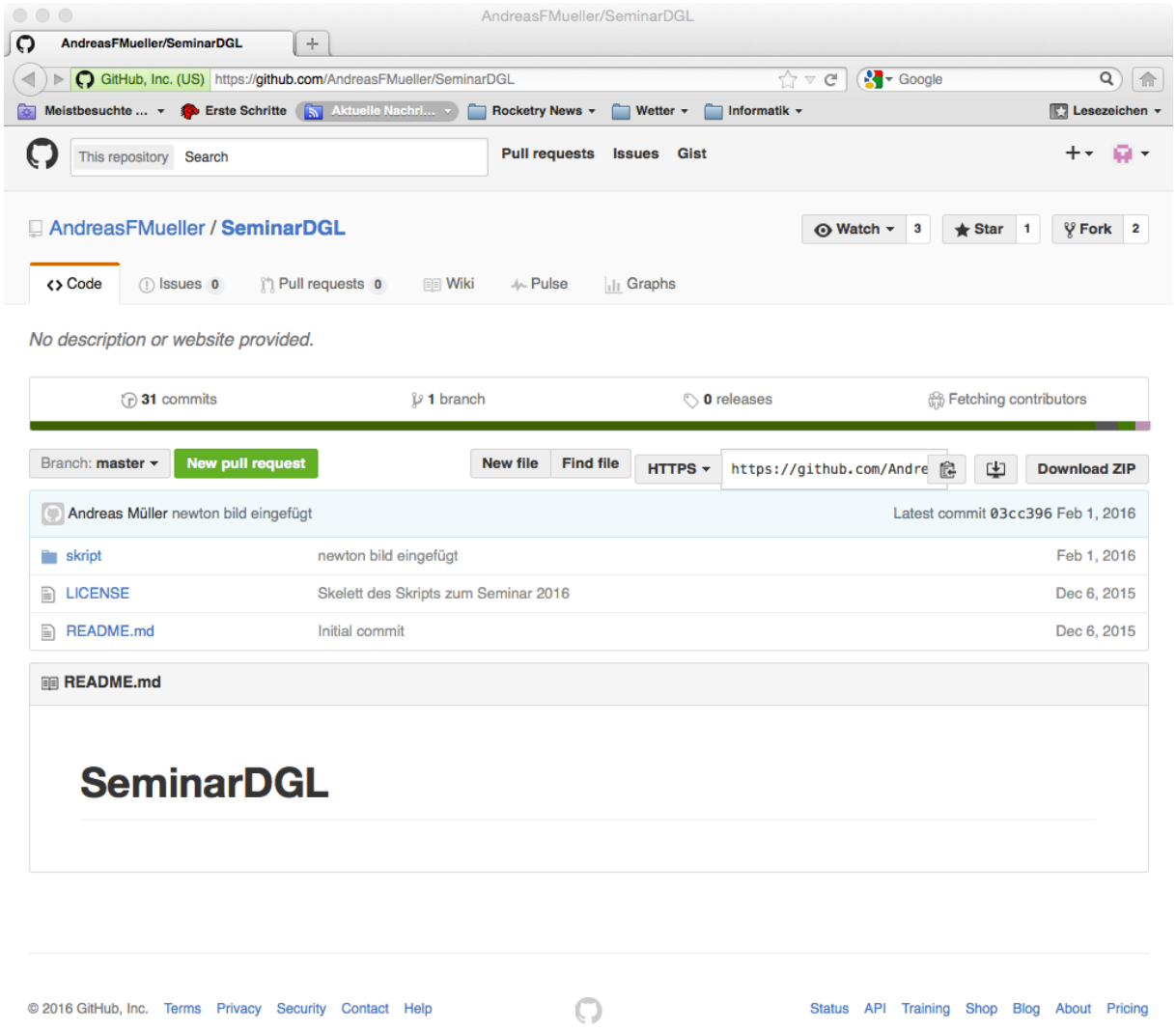


Abbildung 1: Projekt `SeminarDGL` des Benutzers `AndreasFMueller`, bereit zum Forken durch den Benutzer `SeminarTeilnehmer`. Forken erfolgt durch den Knopf **Fork** in der rechten oberen Ecke.

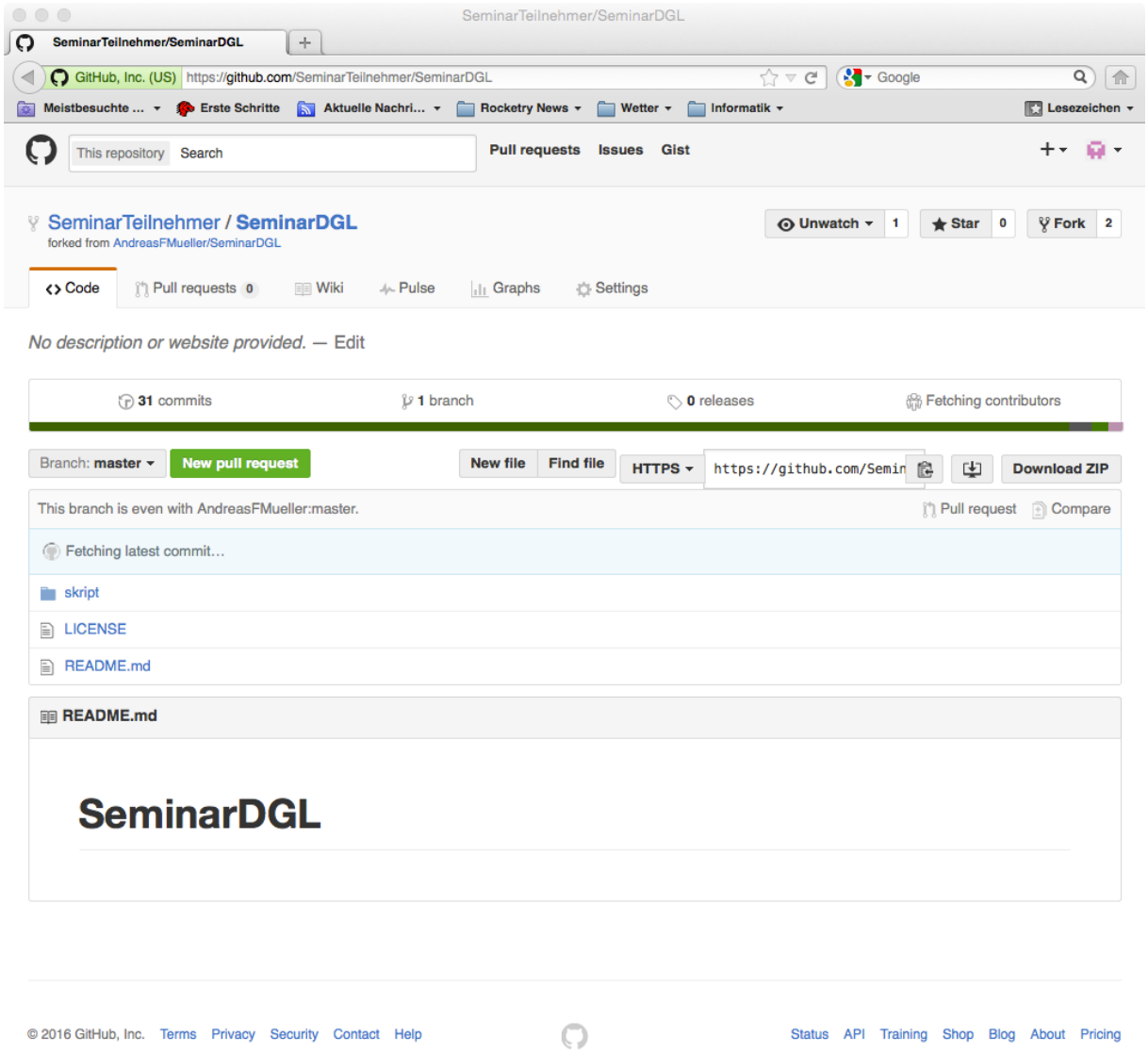


Abbildung 2: Fork des Projektes `SeminarDGL` durch den Benutzer `SeminarTeilnehmer`. An dieser Kopie kann der Benutzer jetzt beliebige Änderungen vornehmen.

Es gibt zwar einen abgekürzten Befehl `commit -a`, der einfach alle Änderungen committet, doch besteht dabei die Gefahr, auch unbeabsichtigte Änderungen aussehalb Ihres eigenen Teilbereiches zu Committen, und damit die Files anderer Projektteilnehmer unbeabsichtigt zu ändern. Sie sollten daher alle Commits immer explizit für die Files eingeben, die Sie tatsächlich geändert haben, also zum Beispiel

```
> git commit einleitung.tex
```

Sie werden dann dazu aufgefordert, einen Kommentar einzugeben. Sie können die Message aber auch gleich mit dem Befehl mitgeben:

```
> git commit -m "neuer Verweis auf Literaturverzeichnis" einleitung.tex
```

Nach diesem Befehl finden Sie ein neues Verzeichnis `SeminarDGL` auf Ihrem Computer, mit allen nötigen Komponenten, um das Skript neu zu bauen. An diesen Files können Sie, mit den genannten Einschränkungen, Ihre Arbeit einfügen und beliebig bearbeiten.

### 3.3.5 Push und Pull

Auch nach einem Commit sind Ihre Änderungen im Repository noch nicht sichtbar, sie sind erst lokal in Ihrem Projekt eingetragen. Der Push-Befehl transferiert Ihre lokalen Änderungen ins Repository, sie werden damit für alle sichtbar. Sie arbeiten nicht allein, möglicherweise hat Ihr Partner Änderungen vorgenommen, die sie auch gerne sehen möchten. Der einfachste Weg ist, diese Änderungen zu “Pushen”, damit sie sichtbar werden. Der Befehl dazu lautet

```
> git push
Password for 'https://SeminarTeilnehmer@github.com':
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 417 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://SeminarTeilnehmer@github.com/SeminarTeilnehmer/SeminarDGL.git
   03cc396..3610eb4  master -> master
```

Sie verwenden dann den Pull-Befehl, um die nun sichtbaren Änderungen in ihre eigene Kopie zu integrieren.

### 3.3.6 Pull-Request

Am Ende möchten Sie Ihre Arbeit in das Gesamtprojekt integrieren. Bis jetzt haben Sie an Ihrem eigenen Clone gearbeitet, der nicht für alle Seminarteilnehmer sichtbar ist. Sie haben auch nicht das Recht, einfach so eine Änderung vorzunehmen, ich als Herausgeber des Seminar-Buches möchte meinen Senf dazu geben können. Sie müssen mir daher Ihren Beitrag mit Hilfe eines Pull-Requests übermitteln. Diesen können Sie direkt auf dem Github Webinterface erzeugen. Ich werde ihn dann daraufhin kontrollieren, ob er die in diesem Dokument beschriebenen Regeln einhält, und in das Projekt integrieren. Sie können auch nach dem Pull-Request weiter an Ihrem Dokument arbeiten, Verbesserungen anbringen. Nutzen Sie die Gelegenheit, schon früh Feedback zu Ihrer Arbeit einzuholen, so ist der Lerneffekt am grössten.

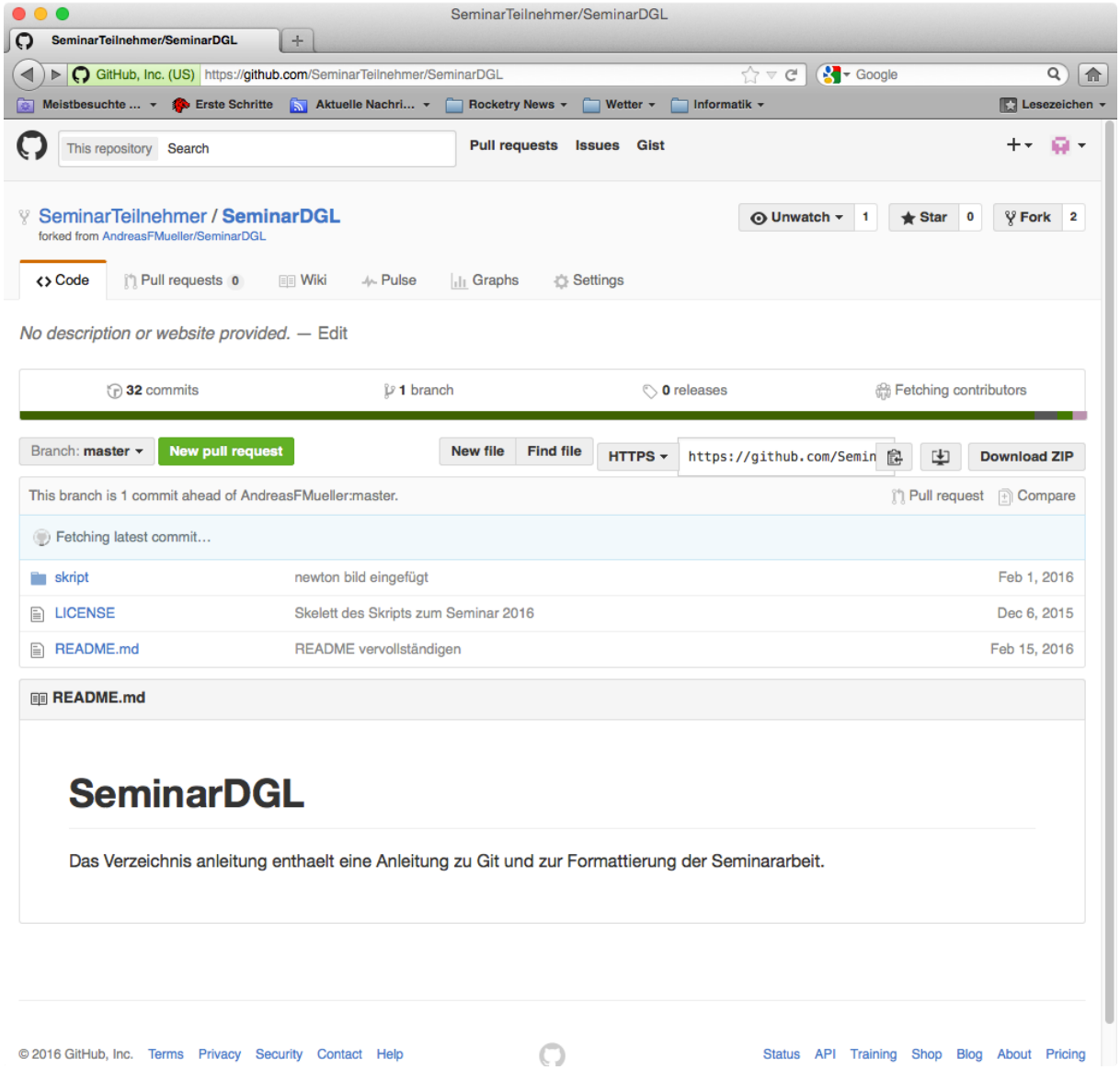


Abbildung 3: Einen neuen Pull-Request erzeugen: Knopf **New pull request** klicken

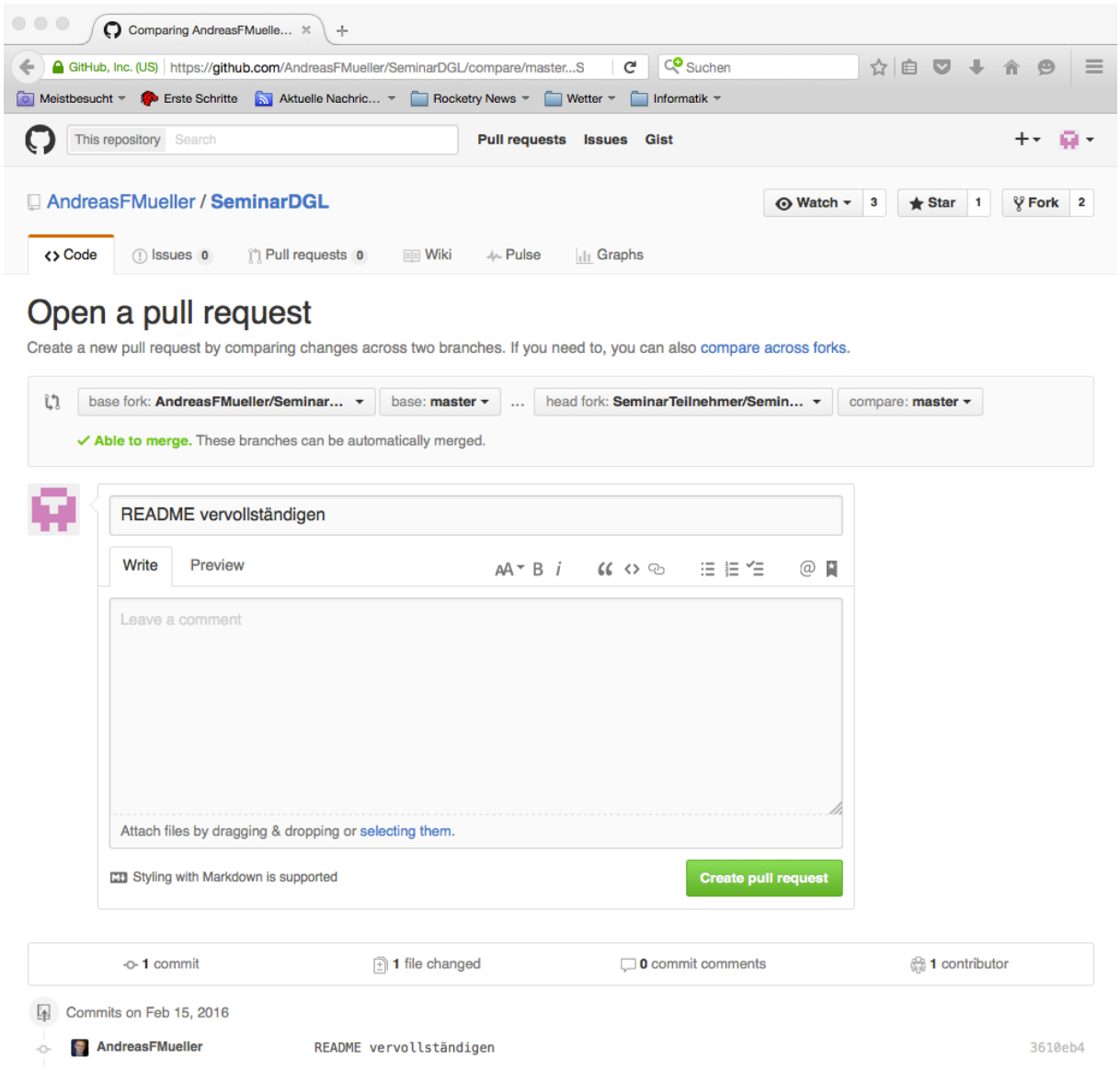


Abbildung 4: Github untersucht, ob der Pull-Request sinnvoll angewendet werden kann, und erlaubt dem Benutzer, einen Kommentar dazu einzugeben.

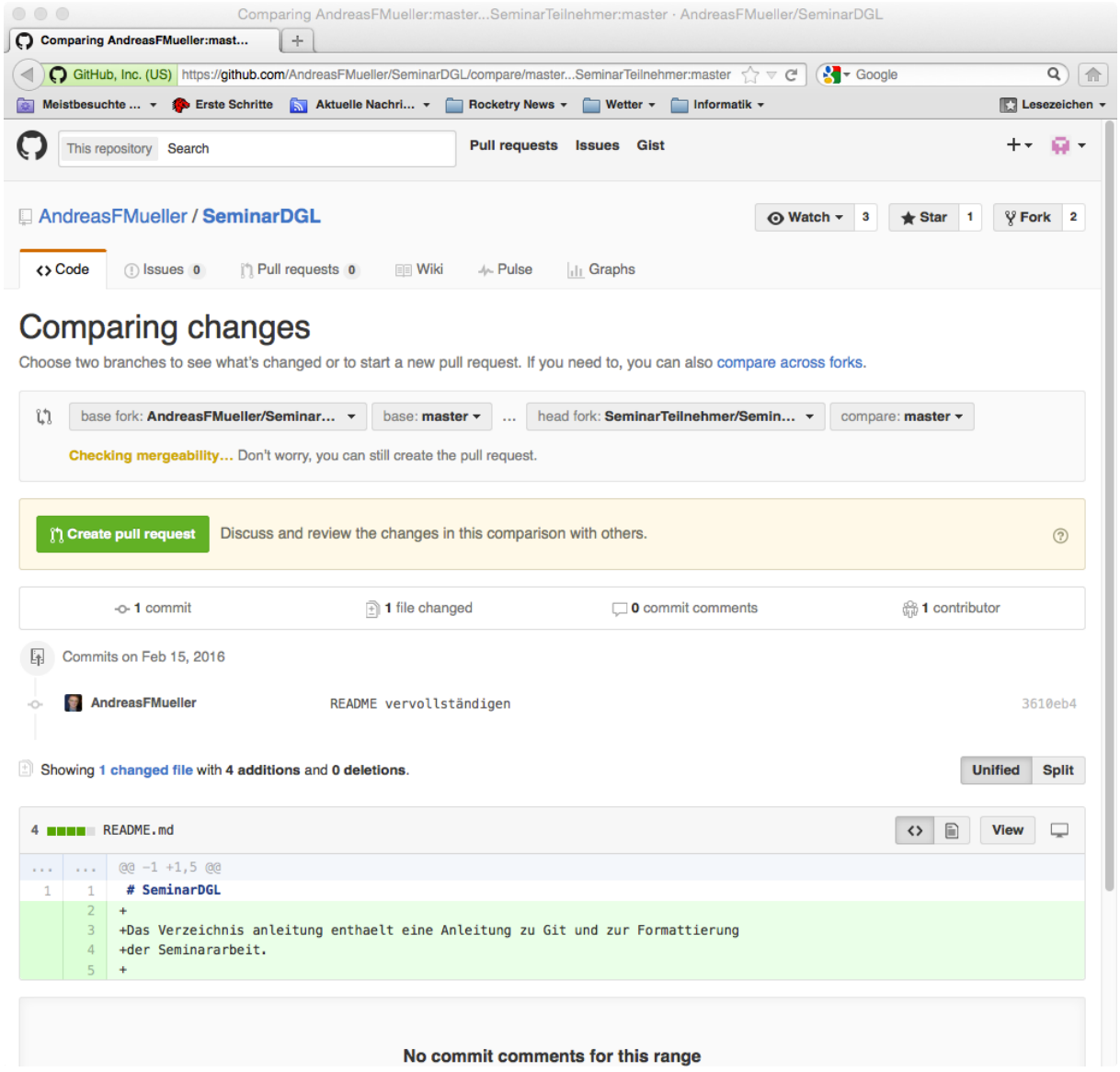


Abbildung 5: Github zeigt die Elemente, die im Pull-Request enthalten sind.

Die Erzeugung eines Pull-Request ist in den Abbildungen 3, 4 und 5 dargestellt. In Abbildung 3 sieht man den Knopf, mit dem der Benutzer `SeminarTeilnehmer` den Pull-Request anfordert. Github untersucht dann, ob der Pull-Request überhaupt sinnvoll erzeugt werden kann, und zeigt die Elemente, die der Pull-Request enthalten wird (Abbildung 4). Der Benutzer kann dann eine Mitteilung eingeben, die dem Verwalter des ursprünglichen Projektes zu verstehen gibt, was der Pull-Request beabsichtigt. In Abbildung 5 ist das Resultat des Pull-Request zu sehen, wie ihn auch der Projektadministrator sieht.

## 3.4 LaTeX

Das TeX-System wurde in den Achtziger-Jahren von Donald Knuth entwickelt, weil er mit der Satz-Qualität nicht zufrieden war, die der Herausgeber seines Buches *The Art of Computer Programming* zu produzieren in der Lage war. Vor allem die Qualität des Formelsatzes war aus Knuths Sicht ungenügend. Knuths System wurde von der Mathematikern und Physikern sehr rasch aufgenommen und weiterentwickelt, besonders in Form des Makropakets LaTeX hat es sich zum Standard für technische Dokumentation entwickelt.

### 3.4.1 Markup

Im Gegensatz zu einem Wysiwyg System wie Microsoft Word verwendet TeX den Markup-Ansatz. Der Autor eines Dokumentes kümmert sich ausschliesslich um den Inhalt, er überlässt die graphische Darstellung überlässt er dem System. Der Input besteht daher aus reinen Text-Files, die neben dem eigentlichen Text einzelne Instruktionen enthalten, die TeX mitteilen, ob es sich bei dem Text um einen Titel, eine Bildunterschrift, oder eine Fussnote handelt. Für die Unterteilung des Textes in Absätze ist überhaupt kein spezieller Markup nötig, eine leere Zeile signalisiert dem TeX-System den Anfang eines neuen Absatzes. Insbesondere ist es falsch, einen Zeilenumbruch mit Hilfe eines doppelten Backslash zu verlangen.

Markup bedeutet auch, dass die Darstellung jederzeit an veränderte Bedingungen angepasst werden kann. Vielleicht muss das Buch bei einem anderen Drucker hergestellt werden, was eine Änderung des Seitenformats zur Folge haben kann. Daher dürfen keine Markup-Instruktionen verwendet werden, die die sich nur auf Aussehen oder Layout beziehen. Zum Beispiel ist es nicht zulässig zu verlangen, dass eine Abbildung an einer ganz bestimmten Stelle auf der Seite erscheint, denn mit anderen Seitenabmessungen könnte diese Anweisung nicht mehr sinnvoll sein.

### 3.4.2 Packages

Packages fassen zusammengehörige Markup-Makros zusammen. Eine breite Auswahl für das Skript nötiger Packages werden durch das Haupt-File `skript.tex` bereits eingelesen. Leider lassen sich nicht alle Package mit jedem anderen Package kombinieren, ausserdem verlängern viele Packages die Zeit, die TeX braucht, um das Skript zusammenzubauen. Sollten Sie für die Darstellung Ihrer Resultate ein Package benötigen, welches noch nicht eingebaut ist, klären Sie bitte erst ab, ob Sie Ihr Ziel auch mit einem vorhandenen Package erreichen können. Falls nicht, fügen Sie das Package zu `skript.tex` hinzu, und kontrollieren Sie, ob dadurch kein Konflikt in anderen Seminararbeiten entsteht. Lässt sich das Skript immer noch erfolgreich bauen, können Sie das neue Package in `skript.tex` Committen und in ihrem Pull-Request dem Projekt hinzufügen.