2 - Einstieg in R

author: Benedict Witzenberger date: 16.04.2019 autosize: true

Was ist R?

> "R is a system for statistical computation and graphics. It consists of a language plus a run-time environment with graphics, a debugger, access to certain system functions, and the ability to run programs stored in script files."
>
> (Quelle: R FAQ)

Was ist R?

- Entstanden 1996 an der University of Auckland
- Entwickelt von Robert Gentleman und Ross Ihaka
- Basiert auf den Sprachen `S` und `Scheme`
- inzwischen gibt es eine breite Entwicklercommunity, die die Sprache voranbringt

R vs Python

R

- Entwickelt von Statistikern
- viele statistische Tools
- einfache Möglichkeiten zur Datenvisualisierung
- gutes User-Interface durch RStudio
- vielfältigere Lösungen für dasselbe Problem
- verbreitet im Journalismus und der Wirtschaft
- relative steile Lernkurve, aber nach den Basics geht es schnell
- Bibliotheken: CRAN (Comprehensive R Archive Network)

Python

- Entwickelt in der Informatik
- multi-funktionelle Sprache
- schneller
- einfachere Syntax
- weniger Lösungen für dasselbe Problem
- relativ flache Lernkurve
- Bibliotheken: PyPi

R vs Python

Es gibt Bibliotheken, die Programmcode der einen Sprache in der anderen Sprache ausführen können.

- R in Python: RPy2
- Python in R: rPython

R vs Python

Fazit:

Beide Sprachen haben vor und Nachteile.

Wir lernen hier R, weil es im Journalismus weiter verbreitet ist. Seine Konzepte stammen eher aus der Statistik stammen - und führen daher für unsere Zwecke schneller zu Ergebnissen.

Heute Vormittag

- Die Kommandozeile
- Editoren
- Github

Die Kommandozeile

Warum?

Jeder nutzt heute hauptsächlich grafische Benutzerinterfaces - aber: über die Kommandozeile geben wir Befehle direkt an den Computer.

Programme können schnell und direkt aufgerufen werden.

Alternative Namen: Konsole, Terminal, Bash, Shell (z.B. bei OS X, Linux)

Microsofts `cmd.exe` nutzt teilweise andere Befehle. Deswegen haben wir versucht, auf Windows 10-Rechnern Linux zu installieren.

Kommandozeile: Wo?

Windows: Windows-Taste, "cmd". Oder Startmenü → Alle Programme → Zubehör → Eingabeaufforderung

Linux Subsystem für Windows: Eingabeaufforderung → "bash"

Mac OS X: Programme → Zubehör → Terminal

Linux: Programme → Zubehör → Terminal (versionsabhängig)

Bash vs DOS

Change Directory: `cd` in Bash, `cd` oder `chdir` in DOS

List Contents of Directory: `ls` in Bash, `dir` in DOS

Move or Rename a File: `mv` in Bash, `move` und `rename` in DOS

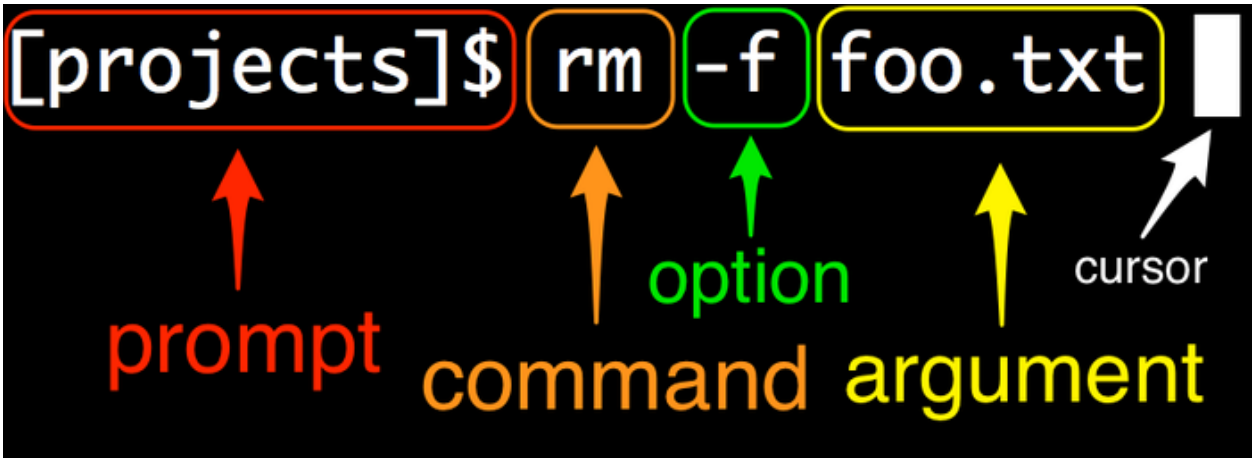
Copy a File: `cp` in Bash, `copy` in DOS

Delete a File: `rm` in Bash, `del` oder `erase` in DOS

Create a Directory: `mkdir` in Bash, `mkdir` in DOS

Use a Text Editor: `vi` or `nano` in Bash, `edit` in DOS

Eine Kommandozeileneingabe



Quelle: https://www.learnenough.com/command-line-tutorial/basics

Let's start!

```
whoami
```

```
benedict@LAPTOP-SG0F5D8L:~$ whoami
benedict
```

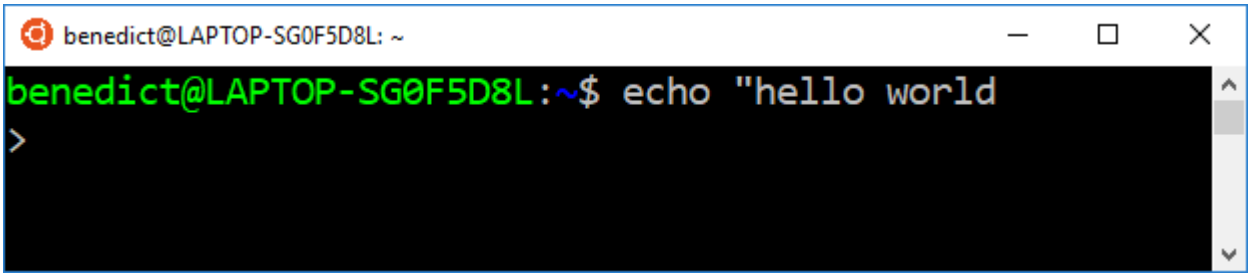
Echo

`echo xyz` gibt *xyz* zurück. In der Konsole ist das vielleicht erstmal egal, aber wir können das auch in Dateien schreiben

```
echo "Hello World"
```

```
echo Hello World
```

```
echo "Hello World
```



Lösung, wenn die Konsole hängt: Strg + C / Ctrl + C

Echo in eine Datei

```
echo "Ich bin ein Test-Text" > test.txt
```

`>` leitet den Inhalt von echo in eine Datei

Wie können wir das überprüfen?

UNIX

`cat`

Windows

`type`

Noch eine Zeile hinzufügen:

```
echo "Ich bin ein weiterer Test-Text" >> test.txt
```

`>>` ist der Append-Operator

Editor öffnen

```
notepad test.txt (Win) | open test.txt (Unix)
```

Verzeichnis wechseln / Dateien anzeigen

UNIX

`cd [Ordnername]` oder `cd ..` für eine Ebene hoch

`ls`

Windows

`cd [Ordnername]` oder `cd ..` für eine Ebene hoch

`dir`

Suchen:

`ls *.txt` oder

`dir *.txt`

Noch detaillierter suchen

UNIX

```
find ./ -iname "datei.endung"
```

```
grep -i -R "suchtext"" ./*
```

WINDOWS

```
dir /b /s "datei.endung"
```

```
findstr /s /m "suchtext"" *.*
```

Hilfe bekommen

UNIX

`help [Kommando]` (auch gut: `man [Kommando]` führt zur Hilfeseite)

WINDOWS

`help [Kommando]`

Meistens funktionieren auch:

```
[Kommando] -h
[Kommando] -help
[Kommando] --help
```

Einen Überblick gewinnen

UNIX

`find .`

`ps -ax`

`kill [PID]`

WINDOWS

`tree`

`tasklist`

`tasklist -PID [PID]`

Neue (leere) Datei erstellen

UNIX

`touch test.txt`

WINDOWS

`echo. > test.txt` oder `type nul > test.txt`

Datei umbenennen

UNIX

`mv test_alt.txt test_neu.txt`

WINDOWS

`rename test_alt.txt test_neu.txt`

Geht auch für alle Dateien mit einer Auswahl:

`mv *.htm *.html`

`rename *.htm *.html`

TIPP

Verwende bei Dateinamen die TAB-Taste für Autovervollständigung.

Datei kopieren

UNIX

```
cp test.txt test_kopie.txt
```

WINDOWS

```
copy test.txt test_kopie.txt
```

Datei löschen

UNIX

`rm test_kopie.txt` (`-f` ohne Nachfragen und `-r` für Unterverzeichnisse)

WINDOWS

`del test_kopie.txt` (`/F` ohne Nachfragen und `/S` für Unterverzeichnisse)

Sehr mächtiges Werkzeug, sollte mit Bedacht eingesetzt werden.

NICHT BENUTZEN: `rm -rf /` im Root-Verzeichnis

TOP DEFINITION

rm -rf /

Finest **compression** avaliable under **UNIX**/Linux! Unfortunatly, there is no decompressor avaliable.

Command issued on **unix** systems to remove a **directory**. If nothing is added after the final slash (/) it will remove the root filesystem. Although, many systems now have protection in place which will report back and say 'rm of / is not allowed' or similar. The command can only be issued by the root user. If you do it as a normal user, it will only remove your home directory and files you have write/execute access to.

*Friend 1: My **hard disk** is full.*
*Friend 2: Type 'rm -rf /'. Its the best **compression** avaliable *evil-grin*.*
*Friend 1: Its **doing it** now, the hard disk is running.*

by Keld July 14, 2005

Zusammen:

1. cd (Win) | pwd (Unix)
2. dir (Win) | ls (Unix)
3. cd Desktop
4. mkdir rBootcamp
5. dir | ls
6. rmdir /S rBootcamp (Win) | rm -r rBootcamp (Unix)
7. exit

Übung Kommandozeile

1. Schreibe "Hallo Welt" in die Konsole.
2. Erstelle ein Verzeichnis höher eine leere Datei mit dem Namen `kommandozeile.txt`.
3. Schreibe "Ich kann Kommandozeile" in die Datei mit dem Namen `kommandozeile.txt`
4. Erstelle eine Kopie von dieser Datei.
5. Überprüfe, ob die Kopie erstellt wurde.
6. Lösche die Kopie über die Kommandozeile.

Weitere nützliche Tools

`ip a` (UNIX) oder `ipconfig` (Windows): Gibt Infos über die aktuellen Netzwerkverbindungen

`curl` kann Dateien herunterladen

`tar` kann Dateien ver- und entpacken

CURL (Client URL)

`curl www.example.com`

`curl -v www.example.com` (Verbose-Mode mit mehr Infos)

`curl -o webseite.html www.example.com`

`curl wttr.in/Munich`

Alternative: wget

Programme installieren

Windows: schwierig

Linux (als Admin): Über apt install

Mac** OS**: Paketmanager Homebrew: https://brew.sh/

Zum Beispiel:

Git: https://wiki.ubuntuusers.de/Git/

```
sudo apt-get install git
```

oder

```
brew install git
```

Beispiel: Bild-Metadaten verifizieren

Exif-Tools als Anhaltspunkte

https://wiki.ubuntuusers.de/ExifTool/

```
exiftool [Bildname]
```

Beispiel: PDFs bearbeiten

pdftk kann PDFs zusammenfügen, teilen, bearbeiten, ausfüllen ...

```
pdftk EINGABEDATEI OPERATION OPTION output AUSGABEDATEI PASSWORT RECHTEOPTION
```

Zusammenfügen:

```
pdftk datei1.pdf datei2.pdf datei3.pdf cat output datei123.pdf
```

Editoren

In der Kommandozeile können wir verschiedene Editoren aufrufen.

UNIX

```
open datei.txt
vim datei.txt
nano datei.txt
```

Windows

```
notepad datei.txt
```

Visuelle Editoren

Für die Softwareentwicklung gibt es gängige Editoren:

Visual Code Studio von Microsoft, Open Source

Sublime sehr schnell, nervt mit Lizenzfrage

Atom Projekt von Github

Unser Glück:

Für R ist die gängige Umgebung RStudio. Theoretisch können wir aber auch einen anderen Editor benutzen.

Git

Versionskontrolle (Version Control System, VCS): Ermöglicht gemeinsames Arbeiten an (Programmier-)Projekten, Nachvollziehen von Arbeitsschritten, Zurücksetzen auf frühere Versionen, Zusammenführen neuer Elemente

Git ist freie Software, sehr populär (Android, Ruby on rails, Eclipse, VLC media player, ...).

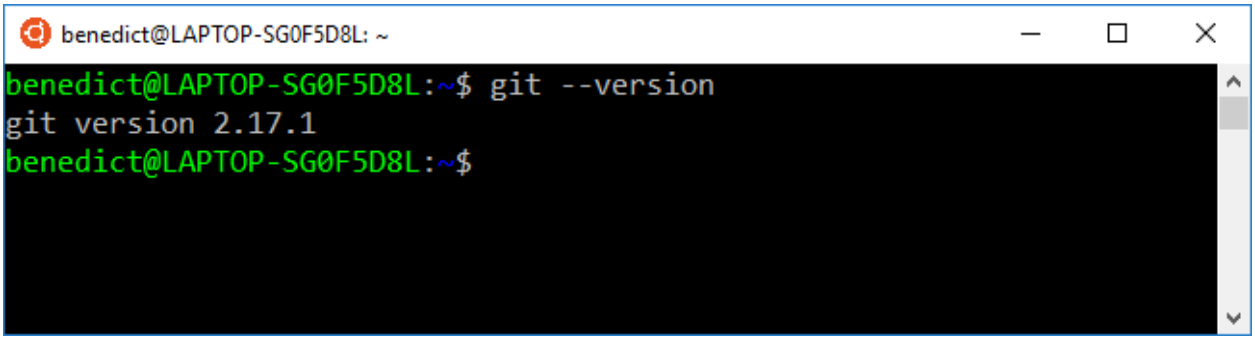
Funktioniert lokal oder auf Servern.

Anbieter z.B: GitHub, GitLab, Bitbucket

Git installieren

Überprüfen wir, welche Version wir installiert haben

```
git --version
```



Installiert haben wir ja gestern schon.

Git konfigurieren

Usernamen eingeben:

```
git config --global user.name "YOUR_USERNAME"
```

Usernamen nochmal checken:

```
git config --global user.name
```

Emailadresse eingeben:

```
git config --global user.email "your_email_address@example.com"
```

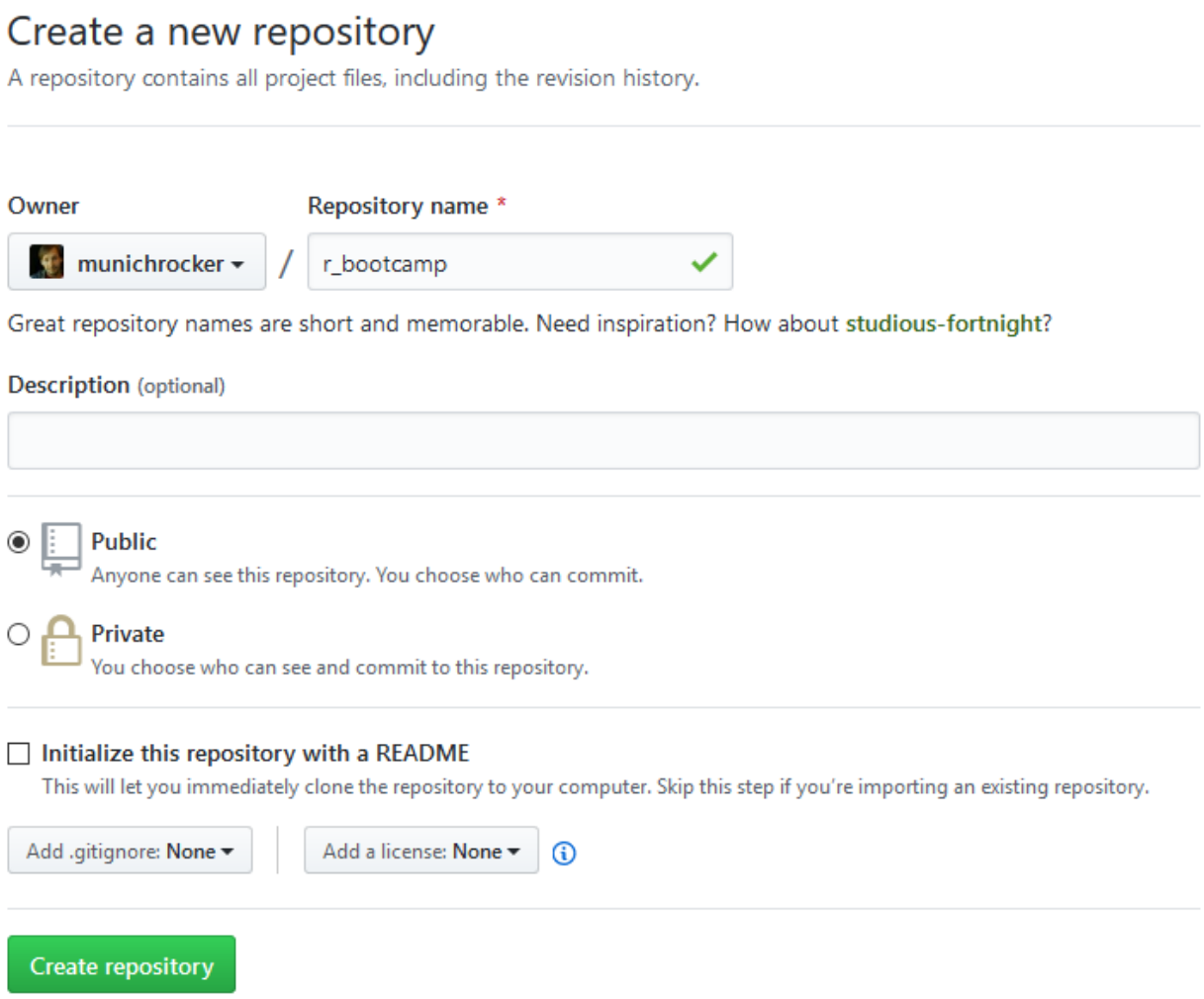
Emailadresse nochmal checken:

```
git config --global user.email
```

Alles checken:

```
git config --global --list
```

Repositories



Branches

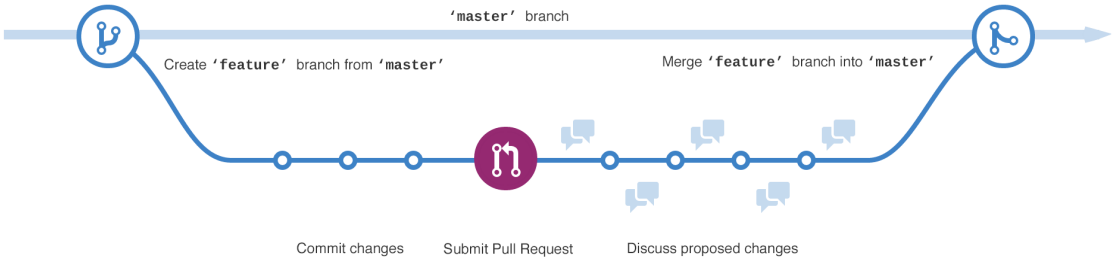
Bekannte Dateinamen sind:

Text.doc

Text-final.doc

Text-final2.doc

Git übernimmt dieses Dilemma für uns mit Branches: neben unserer `master`-Branch können wir beliebig nebenbei arbeiten



Commits

Verschiedene Versionen eins Projekts werden in Commits gespeichert.

Jeder Commit hat eine Commit-Message: zum Beispiel “Intial commit”, “removed xyz”, “added xyz”

Push und Pull

Commitete Dateien, die wir auf unserem Rechner verändert haben, **pushen** wir auf der Git-Server, damit sie für alle Mitarbeitenden verfügbar sind.

Dateien, die auf dem Server verändert sind, **pullen** wir auf unseren Rechner.

Der **Pull-Request** in einem Projekt bedeutet, dass der Code in einem Branch geprüft und in ein Hauptprogejt übernommen werden soll. In Github werden die Unterschiede im Code in rot und grün dargestellt.

Erfolgreiche Pull-Requests werden **gemergt**. Der Code wird zusammengeführt.

Clones

Repositories, in denen wir gerne lokal weiterarbeiten wollen, können wir **klonen**.

Dadurch entsteht eine Verbindung von unserem Computer zum Repository, und wir können immer den aktuellsten Stand des Projekts abrufen.

Die Befehle in Github Bash I

`git init`- Repo im aktuellen Ordner erstellen

`git remote add origin https://github.com/munichrocker/mynewrepository.git` - Repository vom Server verbinden

`git status` - zeigt Veränderungen

`git add .` - fügt alle (durch den Punkt) Dateien im Ordner zu Git hinzu

`git commit -m "Initial Commit"` - fügt die Änderungen zu einem Commit zusammen

Die Befehle in Github Bash II

Branches:

`git branch`

`git checkout -b [neue Branch]` - erstellt neue Branch und wechselt dorthin. Nur Wechseln geht ohne das `-b`

`git push -u origin master` - pusht in die Master-Branch in unserem Server-Repository. Statt Master können wir auch eine andere Branch auswählen.

`git pull origin master` - holt die Dateien vom Server zurück auf den Computer

Für Github gibt es auch grafische Benutzeroberflächen - was euch lieber ist.

Übung Git

1. Erstellt ein neues Repository (privat oder öffentlich) auf Github
2. Klont das Repository auf euren Computer
3. Erstellt eine Textdatei in das lokale Repository
4. Committet und pusht die Datei zu Github
5. Erstellt eine neue Branch und eine neue Datei
6. Pusht die Datei in die neue Branch

Bonus:

7. Versucht auf Github einen Pull-Request zu erstellen und ihn zu mergen.