



# Analyse und Dokumentation

BSc Psychologie SoSe 2024

Belinda Fleischmann

Inhalte basieren teilweise auf Design, Analyse, Dokumentation von Dirk Ostwald, lizenziert unter CC BY-NC-SA 4.0

Datum	Einheit	Thema	Lehrperson
10.04.24	Seminar	(1) Ethik und Ethische Formalitäten	BF
17.04.24	Seminar	(2) Wissenschaftliche Berichte	BF
24.04.24	Seminar	(3) Offenheit und Transparenz	BF
01.05.24	Tag der Arbeit		
<b>08.05.24</b>	<b>Seminar</b>	<b>(4) R, Quarto und Zotero</b>	<b>BF</b>
15.05.24	Praxisseminar	Offene Übung	BF
22.05.24	Präsentationen	Einfache Lineare Regression	JS
29.05.24	Präsentationen	Korrelation	JS
05.06.24	Präsentationen	Einstichproben-T-Test	JS
12.06.24	Präsentationen	Zweistichproben-T-Test	JS
19.06.24	Präsentationen	Einfaktorielle Varianzanalyse	BF
26.06.24	Präsentationen	Zweifaktorielle Varianzanalyse	BF
03.07.24	Präsentationen	Multipe Regression	BF
10.07.24	Präsentationen	Kovarianzanalyse	BF
26.07.24	Klausurtermin		
Feb 2025	Klausurwiederholungstermin		

(4) R, Quarto und Zotero

R Tools

Quarto

Zotero

## **R Tools**

Quarto

Zotero

# Hilfreiche Quellen

---

Visual Studio Code (VS Code) Website

VS Code-R Wiki

R for Data Science (2e)

ggplot2: Elegant Graphics for Data Analysis (3e)

# Wiederholung: R und VS Code



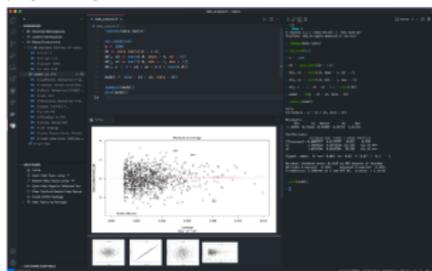
- OVERVIEW
- SETUP
- GET STARTED
- USER GUIDE
- SOURCE CONTROL
- TERMINAL
- GITHUB COPILOT
- LANGUAGES
  - Overview
  - JavaScript
  - JSON
  - HTML
  - CSS, SCSS and Less
  - TypeScript
  - Markdown
  - PowerShell
  - C++
  - Java
  - PHP
  - Python
  - Julia
  - R**
  - Ruby
  - Rust
  - SQL
  - T-SQL
  - C#
  - J#T
  - Fortran
- NODE.JS / JAVASCRIPT
- TEKSCRIPT
- PYTHON
- JAVA
- C++
- C#
- DOCKER
- DATA SCIENCE
- AI/ML

## R in Visual Studio Code

Edit

The **R programming language** is a dynamic language built for statistical computing and graphics. R is commonly used in statistical analysis, scientific computing, machine learning, and data visualization.

The **R extension** for Visual Studio Code supports extended syntax highlighting, code completion, listing, formatting, interacting with R terminals, viewing data, plots, workspace variables, help pages, managing packages and working with **R Markdown** documents.



### IN THIS ARTICLE

- Getting started
- Running R code
- Code completion (IntelliSense)
- Listing
- Workspace views
- Debugging
- Next steps
- Subscribe
- Ask questions
- Follow @code
- Request features
- Report issues
- Watch videos

### Getting started

1. **Install R** (>= 3.4.0) for your platform. For Windows users, it is recommended to check **Save version number in registry** during installation so that the R extension can find the R executable automatically.
2. **Install `languageserver`** in R.  

```
install.packages("languageserver")
```
3. **Install the R extension** for Visual Studio Code.
4. **Create an R file** and start coding.

To enhance the experience of using R in VS Code, the following software and packages are recommended:

VS Code Website

# Wiederholung: R workspace und Interactive Viewer

## Help Viewer

The screenshot displays the Visual Studio Code interface with an R script open. The script defines a function `plot_viewer` that processes data and generates a plot. The Environment Browser shows the current R environment with variables like `data`, `mapping`, `geom_smooth`, and `ggplot1`. The R console shows the execution of the script, including the output of the `geom_smooth` function and the execution of `ggplot1`.

```
R Help - analyse-end-doku-24 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
R
4.R_Quarto_and_Interactive_V
172 # Plot viewer
173 {r echo = T, eval = F}
180
189 # Beschreibung
190 beschreibung <- sprintf(
191   "%r = %2f, beta_0 = %2f, beta_1 = %2f", r, beta_0, beta_1
192 )
193 ...
204
205 # Run Cell Run Above
206 {r echo = T, eval = F}
207 # label: fig-magleichgerade
208 # fig-cap: "Quers der Depressionssymptomatik und Prä-Post-BDI-Differenz."
209 # warning: false
210 library(ggplot2) # Für Pipe (%>),
211 library(ggplot2) # Für ggplot()
212
213 # Daten vorbereiten
214 file_path <- file.path(getwd(), # Pfad zu Daten
215   "4.R_Quarto und Interactive_V", "Daten.csv")
216 D <- read.table(file_path, sep = ";", header = TRUE) # Daten einlesen
217 n_pat <- nrow(D) # Anzahl Patientinnen
218 D_processed <- D %>% # Patientin ID
219
220 # R Console
221 data = D_processed
222 mapping = aes(x = BDI, y = BDI)
223 geom_smooth()
224
225 # Daten
226 # Daten-Axen-Mapping
227 # y-Limits anpassen
228 # Äquivalente zeichnen
229 # Ausgleicherade zeich
230
231 method = "lm",
232 color = "blue", se = F, linewidth = 0.4
233 ) +
234 ylab("BDI Diff") + xlab("Quers Symptomatik (Monate)") # Achsenbeschriftung
235 annotate("text", label = beschreibung) # Zusätzliche Beschreibung
236
237 g <- Inf, hjust = 1, y = 15, vjust = 1)
238 'geom_smooth()' using formula = 'y ~ x'
239
240 # Visualisierung
241 ggplot1(
242   data = D_processed # Daten
243   mapping = aes(x = BDI, y = BDI) # Daten-Axen-Mapping
244 ) + # y-Limits anpassen
245   geom_density(261) # y-Limits anpassen
246   ylab("BDI Diff") + xlab("Quers Symptomatik (Monate)") # Achsenbeschriftung
247
248 # Visualisierung
249 ggplot1(
250   data = D_processed, # Daten
251   mapping = aes(x = BDI, y = BDI) # Daten-Axen-Mapping
252 ) + # y-Limits anpassen
253   geom_density(261) # y-Limits anpassen
254   ylab("BDI Diff") + xlab("Quers Symptomatik (Monate)") # Achsenbeschriftung
255
256 # ViewID_processed()
257 # Showviewer
258 #
```

Mit dem Befehl `?funktionname` über **HELP PAGES** öffnen.

VS Code Wiki - Interactive viewers

# Wiederholung: R workspace und Interactive Viewer

## Table Viewer

The screenshot shows the Visual Studio Code interface with an R script open. The script defines a function `plot.view` that generates a plot and a table. The table viewer displays the output of the function, showing a table with columns `Time`, `TSP`, `DBP`, `DBI`, and `PatientID`.

```
R Code Snippet:  
172 # Plot viewer  
173 --- [r echo = F, eval = F]  
185 # Beschriftung  
206 beschriftung <- sprintf(  
207   "%r = %.2f, beta_0 = %.2f, beta_1 = %.2f", r, beta_0, beta_1  
208 )  
209 ...  
212 # Daten vorbereiten  
213 file_path <- file.path(getwd(),  
214   "4_R_Quarta_und_Zustero", "Daten.csv")  
215 D <- read.table(file_path, sep = ",", header = TRUE) # Daten einlesen  
216 n.pat <- nrow(D) # Anzahl PatientInnen  
217 D_processed <- D %>%  
218   library(ggplot2) # Für ggplot!  
219   library(ggplot2) # Für ggplot!  
220 ...  
225 [r echo = T, eval = F]  
226 # label: fig-angelsichgerade  
227 # fig-cpi: "Dauer der Depressionssymptomatik und Prä-Post DBI-Differenz."  
228 # warning: false  
229 library(ggplot) # Für Pipe (%>),  
230 mutate() # Für ggplot!  
231 library(ggplot2) # Für ggplot!  
232 ...  
233 # Daten vorbereiten  
234 file_path <- file.path(getwd(),  
235   "4_R_Quarta_und_Zustero", "Daten.csv")  
236 D <- read.table(file_path, sep = ",", header = TRUE) # Daten einlesen  
237 n.pat <- nrow(D) # Anzahl PatientInnen  
238 D_processed <- D %>%  
239   library(ggplot2) # Für ggplot!  
240   library(ggplot2) # Für ggplot!  
241 ...  
242 ...  
243 ...  
244 ...  
245 ...  
246 ...  
247 ...  
248 ...  
249 ...  
250 ...  
251 ...  
252 ...  
253 ...  
254 ...  
255 ...  
256 ...  
257 ...  
258 ...  
259 ...  
260 ...  
261 ...  
262 ...  
263 ...  
264 ...  
265 ...  
266 ...  
267 ...  
268 ...  
269 ...  
270 ...  
271 ...  
272 ...  
273 ...  
274 ...  
275 ...  
276 ...  
277 ...  
278 ...  
279 ...  
280 ...  
281 ...  
282 ...  
283 ...  
284 ...  
285 ...  
286 ...  
287 ...  
288 ...  
289 ...  
290 ...  
291 ...  
292 ...  
293 ...  
294 ...  
295 ...  
296 ...  
297 ...  
298 ...  
299 ...  
300 ...  
301 ...  
302 ...  
303 ...  
304 ...  
305 ...  
306 ...  
307 ...  
308 ...  
309 ...  
310 ...  
311 ...  
312 ...  
313 ...  
314 ...  
315 ...  
316 ...  
317 ...  
318 ...  
319 ...  
320 ...  
321 ...  
322 ...  
323 ...  
324 ...  
325 ...  
326 ...  
327 ...  
328 ...  
329 ...  
330 ...  
331 ...  
332 ...  
333 ...  
334 ...  
335 ...  
336 ...  
337 ...  
338 ...  
339 ...  
340 ...  
341 ...  
342 ...  
343 ...  
344 ...  
345 ...  
346 ...  
347 ...  
348 ...  
349 ...  
350 ...  
351 ...  
352 ...  
353 ...  
354 ...  
355 ...  
356 ...  
357 ...  
358 ...  
359 ...  
360 ...  
361 ...  
362 ...  
363 ...  
364 ...  
365 ...  
366 ...  
367 ...  
368 ...  
369 ...  
370 ...  
371 ...  
372 ...  
373 ...  
374 ...  
375 ...  
376 ...  
377 ...  
378 ...  
379 ...  
380 ...  
381 ...  
382 ...  
383 ...  
384 ...  
385 ...  
386 ...  
387 ...  
388 ...  
389 ...  
390 ...  
391 ...  
392 ...  
393 ...  
394 ...  
395 ...  
396 ...  
397 ...  
398 ...  
399 ...  
400 ...  
401 ...  
402 ...  
403 ...  
404 ...  
405 ...  
406 ...  
407 ...  
408 ...  
409 ...  
410 ...  
411 ...  
412 ...  
413 ...  
414 ...  
415 ...  
416 ...  
417 ...  
418 ...  
419 ...  
420 ...  
421 ...  
422 ...  
423 ...  
424 ...  
425 ...  
426 ...  
427 ...  
428 ...  
429 ...  
430 ...  
431 ...  
432 ...  
433 ...  
434 ...  
435 ...  
436 ...  
437 ...  
438 ...  
439 ...  
440 ...  
441 ...  
442 ...  
443 ...  
444 ...  
445 ...  
446 ...  
447 ...  
448 ...  
449 ...  
450 ...  
451 ...  
452 ...  
453 ...  
454 ...  
455 ...  
456 ...  
457 ...  
458 ...  
459 ...  
460 ...  
461 ...  
462 ...  
463 ...  
464 ...  
465 ...  
466 ...  
467 ...  
468 ...  
469 ...  
470 ...  
471 ...  
472 ...  
473 ...  
474 ...  
475 ...  
476 ...  
477 ...  
478 ...  
479 ...  
480 ...  
481 ...  
482 ...  
483 ...  
484 ...  
485 ...  
486 ...  
487 ...  
488 ...  
489 ...  
490 ...  
491 ...  
492 ...  
493 ...  
494 ...  
495 ...  
496 ...  
497 ...  
498 ...  
499 ...  
500 ...  
501 ...  
502 ...  
503 ...  
504 ...  
505 ...  
506 ...  
507 ...  
508 ...  
509 ...  
510 ...  
511 ...  
512 ...  
513 ...  
514 ...  
515 ...  
516 ...  
517 ...  
518 ...  
519 ...  
520 ...  
521 ...  
522 ...  
523 ...  
524 ...  
525 ...  
526 ...  
527 ...  
528 ...  
529 ...  
530 ...  
531 ...  
532 ...  
533 ...  
534 ...  
535 ...  
536 ...  
537 ...  
538 ...  
539 ...  
540 ...  
541 ...  
542 ...  
543 ...  
544 ...  
545 ...  
546 ...  
547 ...  
548 ...  
549 ...  
550 ...  
551 ...  
552 ...  
553 ...  
554 ...  
555 ...  
556 ...  
557 ...  
558 ...  
559 ...  
560 ...  
561 ...  
562 ...  
563 ...  
564 ...  
565 ...  
566 ...  
567 ...  
568 ...  
569 ...  
570 ...  
571 ...  
572 ...  
573 ...  
574 ...  
575 ...  
576 ...  
577 ...  
578 ...  
579 ...  
580 ...  
581 ...  
582 ...  
583 ...  
584 ...  
585 ...  
586 ...  
587 ...  
588 ...  
589 ...  
590 ...  
591 ...  
592 ...  
593 ...  
594 ...  
595 ...  
596 ...  
597 ...  
598 ...  
599 ...  
600 ...  
601 ...  
602 ...  
603 ...  
604 ...  
605 ...  
606 ...  
607 ...  
608 ...  
609 ...  
610 ...  
611 ...  
612 ...  
613 ...  
614 ...  
615 ...  
616 ...  
617 ...  
618 ...  
619 ...  
620 ...  
621 ...  
622 ...  
623 ...  
624 ...  
625 ...  
626 ...  
627 ...  
628 ...  
629 ...  
630 ...  
631 ...  
632 ...  
633 ...  
634 ...  
635 ...  
636 ...  
637 ...  
638 ...  
639 ...  
640 ...  
641 ...  
642 ...  
643 ...  
644 ...  
645 ...  
646 ...  
647 ...  
648 ...  
649 ...  
650 ...  
651 ...  
652 ...  
653 ...  
654 ...  
655 ...  
656 ...  
657 ...  
658 ...  
659 ...  
660 ...  
661 ...  
662 ...  
663 ...  
664 ...  
665 ...  
666 ...  
667 ...  
668 ...  
669 ...  
670 ...  
671 ...  
672 ...  
673 ...  
674 ...  
675 ...  
676 ...  
677 ...  
678 ...  
679 ...  
680 ...  
681 ...  
682 ...  
683 ...  
684 ...  
685 ...  
686 ...  
687 ...  
688 ...  
689 ...  
690 ...  
691 ...  
692 ...  
693 ...  
694 ...  
695 ...  
696 ...  
697 ...  
698 ...  
699 ...  
700 ...  
701 ...  
702 ...  
703 ...  
704 ...  
705 ...  
706 ...  
707 ...  
708 ...  
709 ...  
710 ...  
711 ...  
712 ...  
713 ...  
714 ...  
715 ...  
716 ...  
717 ...  
718 ...  
719 ...  
720 ...  
721 ...  
722 ...  
723 ...  
724 ...  
725 ...  
726 ...  
727 ...  
728 ...  
729 ...  
730 ...  
731 ...  
732 ...  
733 ...  
734 ...  
735 ...  
736 ...  
737 ...  
738 ...  
739 ...  
740 ...  
741 ...  
742 ...  
743 ...  
744 ...  
745 ...  
746 ...  
747 ...  
748 ...  
749 ...  
750 ...  
751 ...  
752 ...  
753 ...  
754 ...  
755 ...  
756 ...  
757 ...  
758 ...  
759 ...  
760 ...  
761 ...  
762 ...  
763 ...  
764 ...  
765 ...  
766 ...  
767 ...  
768 ...  
769 ...  
770 ...  
771 ...  
772 ...  
773 ...  
774 ...  
775 ...  
776 ...  
777 ...  
778 ...  
779 ...  
780 ...  
781 ...  
782 ...  
783 ...  
784 ...  
785 ...  
786 ...  
787 ...  
788 ...  
789 ...  
790 ...  
791 ...  
792 ...  
793 ...  
794 ...  
795 ...  
796 ...  
797 ...  
798 ...  
799 ...  
800 ...  
801 ...  
802 ...  
803 ...  
804 ...  
805 ...  
806 ...  
807 ...  
808 ...  
809 ...  
810 ...  
811 ...  
812 ...  
813 ...  
814 ...  
815 ...  
816 ...  
817 ...  
818 ...  
819 ...  
820 ...  
821 ...  
822 ...  
823 ...  
824 ...  
825 ...  
826 ...  
827 ...  
828 ...  
829 ...  
830 ...  
831 ...  
832 ...  
833 ...  
834 ...  
835 ...  
836 ...  
837 ...  
838 ...  
839 ...  
840 ...  
841 ...  
842 ...  
843 ...  
844 ...  
845 ...  
846 ...  
847 ...  
848 ...  
849 ...  
850 ...  
851 ...  
852 ...  
853 ...  
854 ...  
855 ...  
856 ...  
857 ...  
858 ...  
859 ...  
860 ...  
861 ...  
862 ...  
863 ...  
864 ...  
865 ...  
866 ...  
867 ...  
868 ...  
869 ...  
870 ...  
871 ...  
872 ...  
873 ...  
874 ...  
875 ...  
876 ...  
877 ...  
878 ...  
879 ...  
880 ...  
881 ...  
882 ...  
883 ...  
884 ...  
885 ...  
886 ...  
887 ...  
888 ...  
889 ...  
890 ...  
891 ...  
892 ...  
893 ...  
894 ...  
895 ...  
896 ...  
897 ...  
898 ...  
899 ...  
900 ...  
901 ...  
902 ...  
903 ...  
904 ...  
905 ...  
906 ...  
907 ...  
908 ...  
909 ...  
910 ...  
911 ...  
912 ...  
913 ...  
914 ...  
915 ...  
916 ...  
917 ...  
918 ...  
919 ...  
920 ...  
921 ...  
922 ...  
923 ...  
924 ...  
925 ...  
926 ...  
927 ...  
928 ...  
929 ...  
930 ...  
931 ...  
932 ...  
933 ...  
934 ...  
935 ...  
936 ...  
937 ...  
938 ...  
939 ...  
940 ...  
941 ...  
942 ...  
943 ...  
944 ...  
945 ...  
946 ...  
947 ...  
948 ...  
949 ...  
950 ...  
951 ...  
952 ...  
953 ...  
954 ...  
955 ...  
956 ...  
957 ...  
958 ...  
959 ...  
960 ...  
961 ...  
962 ...  
963 ...  
964 ...  
965 ...  
966 ...  
967 ...  
968 ...  
969 ...  
970 ...  
971 ...  
972 ...  
973 ...  
974 ...  
975 ...  
976 ...  
977 ...  
978 ...  
979 ...  
980 ...  
981 ...  
982 ...  
983 ...  
984 ...  
985 ...  
986 ...  
987 ...  
988 ...  
989 ...  
990 ...  
991 ...  
992 ...  
993 ...  
994 ...  
995 ...  
996 ...  
997 ...  
998 ...  
999 ...  
1000 ...
```

Mit dem Befehl `View()` oder im R **WORKSPACE** → **Global Environment** über das View Symbol  neben entsprechendem Objekt

[VS Code Wiki - Interactive viewers](#)

# R workspace und Interactive Viewer

## List Viewer

The screenshot displays the Visual Studio Code interface with an R script named 'in\_model - analyse-und-doku-24 - Visual Studio Code'. The script defines a linear model and visualizes it. The R code includes:

```
## Plot viewer
plot <- plot.view(
  ~ (r ~ a + b * x, beta_0 = 0.2, beta_1 = 0.2, r, beta_0, beta_1)
)
## Annotieren
plot <- plot.annotate(
  ~ (r ~ a + b * x, beta_0 = 0.2, beta_1 = 0.2, r, beta_0, beta_1)
)
## Daten vorbereiten
file_path <- file.path(getwd(), "Daten")
D <- read.table(file.path, sep = ";", header = TRUE)
n_pat <- nrow(D)
D_processed <- D %>%
  mutate(
    PatientID = seq(n_pat)
  )
## Visualisierung
ggplot(
  data = D_processed,
  mapping = aes(x = DMR, y = BDI)
) +
  coord_cartesian(ylim = c(10, 20)) +
  geom_point() +
  geom_smooth()
## Visualisierung
ggplot(
  data = D_processed,
  mapping = aes(x = DMR, y = BDI)
) +
  coord_cartesian(ylim = c(10, 20)) +
  ylab("BDI Diff") + xlab("Daaver Symptomatik (Monate)")
```

The output pane shows the execution of these commands, including the plot visualization and the resulting R objects like 'D\_processed' and 'ggplot1'.

Mit dem Befehl `View()` oder im R **WORKSPACE** → **Global Environment** über das View Symbol  neben entsprechendem Objekt

[VS Code Wiki - Interactive viewers](#)



# Debugging mit browser()

Die R base Funktion `browser()` erlaubt das Pausieren der Exekution eines Skripts und Inspektion der aktuellen *environment*.

## Beispiel

```
# Beispiel 1
erste_variable <- 1
zweite_variable <- 3
ergebnis <- c()
browser() # Pausiert Skript
print("Das Ergebnis ist: ", ergebnis)
ergebnis <- erste_variable + zweite_variable
browser()

# Beispiel 2
for (i in 1:5) {
  print(i + 2)
  browser()
}
```

Über das Argument `expr` kann auch eine Bedingung als boolesche Operation spezifiziert werden.

Mit `Enter` wird die Exekution fortgeführt.

Mit `Q` wird der browser beendet.

## Motivation

Programmiercode wird streng sequentiell Befehl für Befehl ausgeführt.

Manchmal möchten wir von dieser rein sequentiellen Befehlsreihenfolge abweichen.

Die prinzipiellen Werkzeuge dafür sind **Kontrollstrukturen**. Dazu gehören `if`-statements, `switch`-statements und Schleifen mit `for`, `while` oder `repeat`.

## if-statements

```
if (Bedingung) {  
    TrueAktion      # Befehl, der ausgeführt wird, falls Bedingung TRUE ist  
}
```

- Wenn Bedingung TRUE ist, wird TrueAktion ausgeführt.
- Wenn Bedingung FALSE ist, wird TrueAktion nicht ausgeführt.

## if-else-statements

```
if (Bedingung) {  
    TrueAktion      # Befehl, der ausgeführt wird, falls Bedingung TRUE ist  
} else {  
    FalseAktion     # Befehl, der ausgeführt wird, falls Bedingung FALSE ist  
}
```

- Wenn Bedingung TRUE ist, wird TrueAktion ausgeführt.
- Wenn Bedingung FALSE ist, wird FalseAktion ausgeführt.

### Beispiele

```
x <- 3
if (x > 0) {
  print("x ist größer als 0")
}
```

```
[1] "x ist größer als 0"
```

```
y <- -3
if (y > 0){
  print("y ist größer als 0")
} else{
  print("y ist nicht größer als 0")
}
```

```
[1] "y ist nicht größer als 0"
```

## Wiederholung: Logischer Operatoren

- Die Boolesche Algebra und R kennen zwei *logische Werte*: TRUE und FALSE
- Bei Auswertung von Relationsoperatoren ergeben sich logische Werte

Relationsoperator	Bedeutung
==	Gleich
!=	Ungleich
<, >	Kleiner, Größer
<=, >=	Kleiner gleich, Größer gleich
	ODER
&	UND

- <, <=, >, >= werden zumeist auf numerische Werte angewendet.
- ==, != werden zumeist auf beliebige Datenstrukturen angewendet.
- | und & werden zumeist auf logische Werte angewendet.
- | implementiert das inklusive *oder*. Die Funktion xor() implementiert das exklusive ODER.

# Kontrollstrukturen: if-statements

## Beispiele

```
x <- 3
y <- 2

# Logisches UND/ODER
if (x > 0 | y > 0) {
  print("beide, oder eine der beiden Variablen sind größer 0")
} else{
  print("Keine der Variablen ist größer 0")
}
```

```
[1] "beide, oder eine der beiden Variablen sind größer 0"
```

```
# Logisches UND
if (x > 0 & y > 0) {
  print("x und y sind größer 0")
} else{
  print("Es sind nicht beide Variablen x und y größer 0, aber vielleicht eine der beiden")
}
```

```
[1] "x und y sind größer 0"
```

```
# Exklusives ODER
if (xor(x > 0, y > 0)){
  print("Genau eine der 2 Variablen x und y ist größer 0")
} else{
  print("Es sind entweder keine der Variablen x und y oder beide größer 0")
}
```

```
[1] "Es sind entweder keine der Variablen x und y oder beide größer 0"
```

# Kontrollstrukturen: switch-statements

## Motivation

Kombinierte if-else -statements können leicht unübersichtlich werden.

```
x <- 2
if (x == 1){
  print("Aktion 1")
} else if(x == 2){
  print("Aktion 2")
} else if(x == 3){
  print("Aktion 3")
} else if(x == 4){
  print("Aktion 4")
}
```

```
[1] "Aktion 2"
```

## switch-statement mit Integer

```
x <- 2
switch(
  x,                                # switch Variable
  print("Aktion 1"),                # 1. Aktion
  print("Aktion 2"),                # 2. Aktion
  print("Aktion 3"),                # 3. Aktion
  print("Aktion 4")                 # 4. Aktion
)
```

```
[1] "Aktion 2"
```

### switch-statement mit Character

```
x <- "a"
switch(
  x,                                # switch Variable
  a = print("Aktion 1"),            # 1. Aktion
  b = print("Aktion 2"),            # 2. Aktion
  c = print("Aktion 3"),            # 3. Aktion
  d = print("Aktion 4")             # 4. Aktion
)
```

```
[1] "Aktion 1"
```

## for-Schleifen

```
for (item in sequenz){  
    zu_wiederholende_Aktion          # Aktion, die wiederholt werden soll  
}
```

## Beispiel

```
for (i in 1:3) {  
    print(i)                          # Aktion, die wiederholt werden soll  
}
```

```
[1] 1  
[1] 2  
[1] 3
```

## while-Schleifen

while-Schleifen iterieren Codeabschnitte basierend auf einer Bedingung.

```
while (Bedingung) {  
  TrueAktion          # TrueAktion wird ausgeführt, solange Condition == TRUE  
}
```

Beispiel

```
i <- 5  
while (i < 11) {  
  print(i)  
  i <- i + 1  
}
```

```
[1] 5  
[1] 6  
[1] 7  
[1] 8  
[1] 9  
[1] 10
```

## repeat-Schleifen

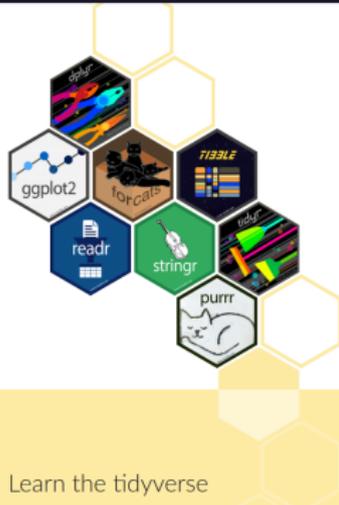
repeat-loops wiederholen Codeabschnitte bis zu einem 'break' Befehl

```
repeat {  
  TrueAktion          # Aktion wird ausgeführt, bis ein break Befehl evaluiert wird  
}
```

### Beispiel

```
i <- 1  
repeat {  
  print(i)  
  i <- i + 1  
  if (i == 5) {  
    break  
  }  
}
```

```
[1] 1  
[1] 2  
[1] 3  
[1] 4
```



## R packages for data science

The tidyverse is an opinionated **collection of R packages** designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

Install the complete tidyverse with:

```
install.packages("tidyverse")
```

Learn the tidyverse

[Tidyverse](#)

[Cheat Sheets](#)

## Data transformation with dplyr : : CHEATSHEET



dplyr functions work with pipes and expect **tidy data**. In tidy data:

<pre>mpg  &gt; summarise(   mean_mpg = mean(mpg))</pre>	<pre>mpg  &gt; summarise(   mean_mpg = mean(mpg))</pre>	<p><b>pipes</b></p> <p>Each <b>variable</b> is in its own <b>column</b></p> <p>Each <b>observation, or case</b>, is in its own <b>row</b></p> <p><b>x</b>  &gt; <b>f(y)</b> becomes <b>f(x, y)</b></p>
---	---	--

### Summarize Cases

Apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).

<pre>mpg  &gt; summarise(   mean_mpg = mean(mpg))</pre>	<p><b>summary function</b></p>
<pre>mpg  &gt; summarise(   mean_mpg = mean(mpg))</pre>	<p>Compute table of summaries.</p>
<pre>mpg  &gt; summarise(   mean_mpg = mean(mpg))</pre>	<p><b>count()</b> Count number of rows in each group defined by the variables in ... Also <b>tally()</b>, <b>add_count()</b>, <b>add_tally()</b></p>
<pre>mpg  &gt; summarise(   count_per_cyl = count(cyl))</pre>	<p><b>counting()</b></p>

### Group Cases

Use **group\_by()** to create a "grouped" copy of a table grouped by columns in ... dplyr functions will manipulate each "group" separately and combine the results.

<pre>mpg  &gt; summarise(   mean_mpg = mean(mpg))</pre>	<pre>mpg  &gt; summarise(   mean_mpg = mean(mpg))</pre>
<pre>mpg  &gt; summarise(   mean_mpg = mean(mpg))</pre>	<p>See <b>7base:Logic and Comparison</b> for help.</p>

Use **rowwise()** to group data into individual rows. dplyr functions will compute results for each row. Also apply functions to list-columns. See tidy cheat sheet for list-column workflow.

<pre>starwars  &gt; rowwise()  &gt; mutate(   film_count = length(films))</pre>	<p><b>starwars</b>  &gt; <b>rowwise()</b>  &gt; <b>mutate(film_count = length(films))</b></p>
---	---

**ungroup()** Returns ungrouped copy of table.

```
ungroup(mtcars)
```



### Manipulate Cases

#### EXTRACT CASES

Row functions return a subset of rows as a new table.

<pre>mpg  &gt; filter(mpg &gt; 20)</pre>	<p><b>filter()</b> Extract rows that meet logical criteria.</p>
<pre>mpg  &gt; distinct(mpg)</pre>	<p><b>distinct()</b> Remove rows with duplicate values.</p>
<pre>mpg  &gt; slice(10:15)</pre>	<p><b>slice()</b> Select rows by position.</p>
<pre>mpg  &gt; slice_sample(n = 5)</pre>	<p><b>slice_sample()</b> Randomly select rows. Use <b>n</b> to select a number of rows and <b>prop</b> to select a fraction of rows.</p>
<pre>mpg  &gt; slice_min(mpg)</pre>	<p><b>slice_min()</b> and <b>slice_max()</b> Select rows with the lowest and highest values.</p>
<pre>mpg  &gt; slice_head(mpg)</pre>	<p><b>slice_head()</b> and <b>slice_tail()</b> Select the first or last rows.</p>

**Logical and boolean operators to use with filter()**

```
== < <= > >= is.na() %in% | xor()
! & && &&& &&&&
```

#### ARRANGE CASES

<pre>mpg  &gt; arrange(mpg)</pre>	<p><b>arrange()</b> Order rows by values of a column or columns (low to high), use <b>with_desc()</b> to order from high to low.</p>
-----------------------------------	--

#### ADD CASES

<pre>cars  &gt; add_row(speed = 1, dist = 1)</pre>	<p><b>add_row()</b> Add one or more rows to a table.</p>
--	--

### Manipulate Variables

#### EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.

<pre>mtcars  &gt; pull(wt)</pre>	<p><b>pull()</b> Extract column values as a vector, by name or index.</p>
<pre>mtcars  &gt; select(mpg, wt)</pre>	<p><b>select()</b> Extract columns as a table.</p>
<pre>mtcars  &gt; relocate(mpg, cyl)</pre>	<p><b>relocate()</b> Move columns to new position.</p>

#### Use these helpers with select() and across()

**contains()**, **ends\_with()**, **starts\_with()**, **matches()**, **num\_range()**, **all\_of()**, **any\_of()**, **everything()**

#### MANIPULATE MULTIPLE VARIABLES AT ONCE

<pre>df  &gt; across(cyl, disp, wt, hp, qsec, gear, carb)</pre>	<p><b>across()</b> Summarize or mutate multiple columns in the same way.</p>
<pre>df  &gt; summarise(across(everything(), mean))</pre>	<p><b>summarise(across())</b> Compute across columns in row-wise data.</p>

#### MAKE NEW VARIABLES

Apply **vectorized functions** to columns. Vectorized functions take vectors as input and return vectors of the same length as output (see back).

<pre>mtcars  &gt; mutate(gpm = 1 / mpg)</pre>	<p><b>mutate()</b> Compute new column(s). Also <b>add_column()</b>.</p>
<pre>mtcars  &gt; rename(gpm = 1 / mpg)</pre>	<p><b>rename()</b> Rename columns. Use <b>rename_with()</b> to rename with a function.</p>

# Datenvorverarbeitung mit dplyr

```
D <- read.table("Daten_1.csv", sep = ",", header = TRUE) # Daten einlesen
```

Variable_1	Variable_2	Variable_3
34.87	34.61	33.56
32.16	22.89	15.75
33.95	31.82	28.83
28.78	25.91	20.04
30.13	26.83	22.00
30.50	26.50	24.42
32.48	26.92	22.96
31.66	31.84	28.83
32.76	33.00	33.28
31.60	26.77	21.21
32.44	28.55	28.63
29.48	25.33	24.19
31.24	28.97	25.18
34.33	31.31	28.22
31.56	27.11	22.92
31.87	30.95	30.30
27.07	21.94	17.60
29.36	25.41	19.32
36.07	33.56	33.41
33.03	28.81	26.58
33.12	32.20	29.44

# Datenvorverarbeitung mit dplyr

Der Pipe operater %>% oder |> ermöglicht es, Funktionen in einer Reihe nacheinander auszuführen.

Mit der R-Funktion mutate() können wir neue Spalten erzeugen (auch als Funktionen bestehender Spalten).

```
library(dplyr)
n <- nrow(D)
D_processed <- D %>%
  mutate(ID = seq(n)) %>%
  mutate(Summe = Variable_1 + Variable_2 + Variable_3)
```

# Anzahl Beobachtungen  
# D wird an nächste Funktion übergeben  
# ID-Spalte hinzufügen  
# Summen-Spalte hinzufügen

Variable_1	Variable_2	Variable_3	ID	Summe
34.87	34.61	33.56	1	103.04
32.16	22.89	15.75	2	70.79
33.95	31.82	28.83	3	94.60
28.78	25.91	20.04	4	74.74
30.13	26.83	22.00	5	78.96
30.50	26.50	24.42	6	81.42
32.48	26.92	22.96	7	82.37
31.66	31.84	28.83	8	92.34
32.76	33.00	33.28	9	99.05
31.60	26.77	21.21	10	79.58
32.44	28.55	28.63	11	89.62
29.48	25.33	24.19	12	79.00
31.24	28.97	25.18	13	85.40
34.33	31.31	28.22	14	93.86
31.56	27.11	22.92	15	81.59
31.87	30.95	30.30	16	93.12
27.07	21.94	17.60	17	66.61
29.36	25.41	19.32	18	74.09
36.07	33.56	33.41	19	103.04
33.03	28.81	26.58	20	88.42
33.12	32.20	29.44	21	94.75

# Datenvorverarbeitung mit dplyr

Mit der R-Funktion `filter()` können wir Zeilen gemäß bestimmten Bedingungen auswählen.

```
D_selected <- D_processed %>%  
  filter(ID %in% 1:10) %>%           # Auswahl der IDs 1-10  
  filter(Summe > 90)                # Selektion der Beobachtungen mit Summe > 90
```

Variable_1	Variable_2	Variable_3	ID	Summe
34.87	34.61	33.56	1	103.04
33.95	31.82	28.83	3	94.60
31.66	31.84	28.83	8	92.34
32.76	33.00	33.28	9	99.05

## Data visualization with ggplot2 : : CHEATSHEET



### Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data set**, a **coordinate system**, and **geoms**—visual marks that represent data points.



To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.



Complete the template below to build a graph.

```
ggplot(data = <DATA> +
  <GEOM FUNCTION> mapping = aes(<MAPPING>,
  <COORDINATE SYSTEM> + <POSITION> +
  <COORDINATE SYSTEM> +
  <FACE> FUNCTION +
  <SCALE FUNCTION> +
  <THEME FUNCTION>)
```

**ggplot**(data = mpg, aes(x = cty, y = hwy)) Begins a plot that you finish by adding layers to. Add one geom function per type.

**last\_plot()** Returns the last plot.

**ggsave**("plot.png", width = 5, height = 5) Saves last plot as 5" x 5" file named "plot.png" in working directory. Matches file type to file extension.

### Aes

Common aesthetic values. **color** and **fill** - string ("red", "#990000").

**linetype** - integer or string (0 = "blank", 1 = "solid", 2 = "dashed", 3 = "dotted", 4 = "dotteddash", 5 = "longdash", 6 = "twodash").

**size** - integer (line width in mm)

**shape** - integer(shape name or a single character ("a"))



### Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

#### GRAPHICAL PRIMITIVES

```
a <- ggplot(economics, aes(date, unempLOY))
b <- ggplot(mpg, aes(x = long, y = lat))

a + geom_blank() and a + expand_limits()
ensure limits outside values across all plots.

b + geom_curve(aes(x = long, y = lat,
  send = long + 1, curvature = 1) - x, send, y, yend,
  alpha, angle, color, curvature, linetype, size)

a + geom_path(linestring = "bezier",
  linestring = "round", linestring = 2)
x, y, alpha, color, group, linetype, size

a + geom_polygon(aes(alpha = 50)) - x, y, alpha,
  color, fill, group, subgroup, linetype, size

b + geom_rect(aes(xmin = long, ymin = lat,
  xmax = long + 2, ymax = lat + 2) - xmin, xmin,
  ymax, ymin, alpha, color, fill, linetype, size)

a + geom_ribbon(aes(ymin = unempLOY - 900,
  ymax = unempLOY + 900)) - x, ymax, ymin,
  alpha, color, fill, group, linetype, size
```

#### LINE SEGMENTS

```
common aesthetics: x, y, alpha, color, linetype, size

b + geom_abline(aes(intercept = 0, slope = 1))
a + geom_hline(aes(yintercept = lat))
b + geom_vline(aes(xintercept = long))

b + geom_segment(aes(yend = lat + 1, send = long + 1))
a + geom_spoke(aes(angle = 1:155, radius = 1))
```

#### ONE VARIABLE continuous

```
c <- ggplot(mpg, aes(hwy))
c2 <- ggplot(mpg)

c + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size

c + geom_density(kernel = "gaussian")
x, y, alpha, color, fill, group, linetype, size, weight

c + geom_dotplot()
x, y, alpha, color, fill

c + geom_freqpoly()
x, y, alpha, color, fill, group, linetype, size

c + geom_histogram(binwidth = 3)
x, y, alpha, color, fill, linetype, size, weight

c2 + geom_qq(aes(sample = hwy))
x, y, alpha, color, fill, linetype, size, weight
```

#### discrete

```
d <- ggplot(mpg, aes(f))

d + geom_bar()
x, alpha, color, fill, linetype, size, weight
```

#### TWO VARIABLES both continuous

```
a + geom_label(aes(label = cty, nudje_x = 1,
  nudje_y = 2) - x, y, label, alpha, angle, color,
  family, fontface, fjust, fontweight, size, vjust)

a + geom_point()
x, y, alpha, color, fill, shape, size, stroke

a + geom_quantile()
x, y, alpha, color, group, linetype, size, weight

a + geom_rug(sides = "tl")
x, y, alpha, color, linetype, size

a + geom_smooth(method = lm)
x, y, alpha, color, fill, group, linetype, size, weight

a + geom_text(aes(label = cty, nudje_x = 1,
  nudje_y = 2) - x, y, label, alpha, angle, color,
  family, fontface, font, fontweight, size, vjust)
```

#### one discrete, one continuous

```
f <- ggplot(mpg, aes(class, hwy))

f + geom_col()
x, y, alpha, color, fill, group, linetype, size

f + geom_boxplot()
x, y, lower, middle, upper, ymax, xmin, alpha,
  color, fill, group, linetype, shape, size, weight

f + geom_dotplot(binwidth = "y", stackdir = "center")
x, y, alpha, color, fill, group

f + geom_violin(scale = "area")
x, y, alpha, color, fill, group, linetype, size, weight
```

#### both discrete

```
g <- ggplot(diamonds, aes(carat, color))

g + geom_count()
x, y, alpha, color, fill, shape, size, stroke

g + geom_jitter(height = 2, width = 2)
x, y, alpha, color, fill, shape, size
```

#### THREE VARIABLES

```
h <- ggplot(diamonds, aes(carat, log2(depth), fill))
h + geom_contour(geom = "rect")
x, y, alpha, color, group, linetype, size, weight

i + geom_contour_filled(aes(fill = z))
x, y, alpha, color, fill, group, linetype, size, subgroup
```

#### continuous bivariate distribution

```
h <- ggplot(diamonds, aes(carat, price))

h + geom_bin2d(binwidth = c(0.25, 500))
x, y, alpha, color, fill, linetype, size, weight

h + geom_density_2d()
x, y, alpha, color, group, linetype, size

h + geom_hex()
x, y, alpha, color, fill, size
```

#### continuous function

```
i <- ggplot(economics, aes(date, unempLOY))

i + geom_area()
x, y, alpha, color, fill, linetype, size

i + geom_line()
x, y, alpha, color, group, linetype, size

i + geom_step(direction = "hv")
x, y, alpha, color, group, linetype, size
```

#### visualizing error

```
df <- data.frame mpg = c("A", "B"), fit = 45, se = 12)
j <- ggplot(df, aes(mpg, fit, ymin = fit - se, ymax = fit + se))

j + geom_crosstab(fit = 2) - x, y, ymax,
  ymin, alpha, color, fill, group, linetype, size

j + geom_errorbar() - x, ymax, ymin,
  alpha, color, group, linetype, size, width
Also geom_errorbarh()

j + geom_linerange()
x, y, ymin, ymax, alpha, color, fill, group, linetype, size

j + geom_pointrange() - x, y, ymin, ymax,
  alpha, color, fill, group, linetype, shape, size
```

#### maps

```
maps <- data.frame(murder = USState$Murder,
  state = tolower(row.names(USState)))
map <- map_data("state")

k <- ggplot(maps, aes(lon, lat, murder))

k + geom_map(map_id = state, map = map)
+ expand_limits(x = mapping$lon, y = map$lat)
map_id, alpha, color, fill, linetype, size
```

# Plotten mit ggplot2

## Beispieldatensatz

```
library(dplyr) # Für Pipe (%>%), mutate()

# Daten vorbereiten
D <- read.table("Daten_2.csv", sep = ",", header = TRUE) # Daten einlesen
n_pat <- nrow(D) # Anzahl Patientinnen
D_processed <- D %>% # PatientIn ID hinzufügen
  mutate(PatientIn = seq(n_pat))
```

Die ersten 12 Zeilen des Dataframes:

DUR	BDI	PatientIn
1.37	9	1
2.18	8	2
1.16	11	3
3.60	0	4
2.33	8	5
1.18	7	6
2.49	3	7
2.74	7	8
2.58	3	9
1.69	11	10
3.51	9	11
2.39	7	12

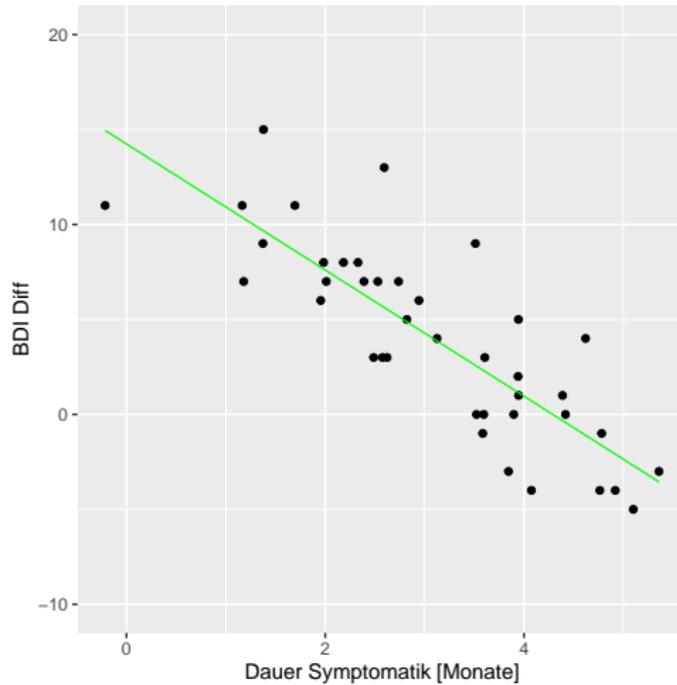
# Plotten mit ggplot2

```
library(ggplot2) # Für ggplot()

# Visualisierung
ggplot(
  data = D_processed, # Daten
  mapping = aes(x = DUR, y = BDI) # Daten-Axen-mapping
) +
  coord_cartesian(ylim = c(-10, 20)) + # y-limits anpassen
  geom_point() + # Datenpunkte zeichnen
  geom_smooth( # Ausgleichsgerade zeichnen
    method = "lm",
    color = "green", se = F, linewidth = 0.4
  ) +
  ylab("BDI Diff") + xlab("Dauer Symptomatik [Monate]") # Achsenbeschriftung
graphics.off() # Schließt browser

ggsave( # Abbildung speichern
  filename = "ggplot_beispiel.pdf",
  height = 5, width = 5
)
```

# Plotten mit ggplot2



R Tools

**Quarto**

Zotero

## Welcome to Quarto

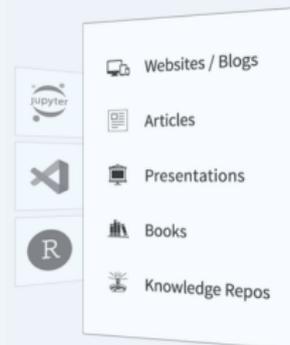
### An open-source scientific and technical publishing system

- Author using [Jupyter](#) notebooks or with plain text markdown in your favorite editor.
- Create dynamic content with [Python](#), [R](#), [Julia](#), and [Observable](#).
- Publish reproducible, production quality articles, presentations, websites, blogs, and books in HTML, PDF, MS Word, ePub, and more.
- Share knowledge and insights organization-wide by publishing to [Posit Connect](#), [Confluence](#), or other publishing systems.
- Write using [Pandoc](#) markdown, including equations, citations, crossrefs, figure panels, callouts, advanced layout, and more.

**Analyze. Share. Reproduce. You have a story to tell with data—tell it with Quarto.**

Get Started

Guide



Quarto Website

## Was ist Quarto?

- Ein seit 2022 verfügbares freies wissenschaftlich-technisches Publikationssystem
- Eine Weiterentwicklung von [RMarkdown](#) und [RBookdown](#) durch [Posit](#)
- RMarkdown/RBookdown sind RStudio Adaptationen von [Markdown](#) und [Jupyter Notebooks](#)
- Allgemeines Ziel ist hier die einfache Integration von ausführbarem Programmiercode in ein ansprechendes Text-, Tabellen- und Abbildungslayout für Web- und Printdokumente.
- Quarto nutzt [Markdown](#) und [Latex](#) für Layoutprozesse.
- Quarto nutzt [Pandoc](#) für multiple Outputformate (.html, .docx, .pdf, etc.)
- Quarto läuft smoother und schneller als RMarkdown und RBookdown.



## Get Started

Tutorial: Hello, Quarto  
Tutorial: Computations  
Tutorial: Authoring

## Get Started

Install Quarto, then check out the tutorials to learn the basics.

### Step 1

Install Quarto

Find your operating system in the table below

Platform	Download	Size	SHA-256
Ubuntu 18+/Debian 10+	<a href="#">quarto-1.4.554-linux-amd64.deb</a>	111.82 MB	7b57a62
Linux x86 Tarball	<a href="#">quarto-1.4.554-linux-amd64.tar.gz</a>	113.04 MB	f01283f
Linux Arm64	<a href="#">quarto-1.4.554-linux-arm64.deb</a>	112.52 MB	4201a1b
Linux Arm64 Tarball	<a href="#">quarto-1.4.554-linux-arm64.tar.gz</a>	113.6 MB	43c788a
RHEL 7 Tarball	<a href="#">quarto-1.4.554-linux-rhel7-amd64.tar.gz</a>	113.4 MB	7d5264b
Mac OS	<a href="#">quarto-1.4.554-macos.pkg</a>	186.2 MB	ab6a44c
Windows	<a href="#">quarto-1.4.554-win.msi</a>	108.89 MB	f6d281d

[Release notes and more downloads...](#)

### Step 2

Choose your tool and get started



Quarto Website Installation

 Overview Get Started Guide Extensions Reference Gallery Blog Help 

Get Started  
Tutorial: Hello, Quarto  
Tutorial: Computations  
Tutorial: Authoring

## Tutorial: Hello, Quarto

Choose your tool

 VS Code  Jupyter  RStudio  Neovim  Editor

### Overview

In this tutorial we'll show you how to use Quarto with VS Code. Before getting started, you should install the [Quarto VS Code Extension](#), which includes many tools that enhance working with Quarto, including:

- Integrated render and preview for Quarto documents.
- Syntax highlighting for markdown and embedded languages
- Completion and diagnostics for YAML options
- Completion for embedded languages (e.g. Python, R, Julia, etc.)
- Commands and key-bindings for running cells and selected lines.

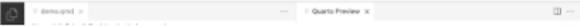
You can install the Quarto extension from within the **Extensions** tab in VS Code, from the [Extension Marketplace](#), the [Open VSX Registry](#) or directly from a [VIX extension file](#).

**Note**

This tutorial focuses on editing plain text Quarto `.qmd` files in VS Code. Depending on your preferences and the task at hand there are two other editing modes available for Quarto documents: the [Visual Editor](#) and the [Notebook Editor](#). For the purposes of learning we recommend you work through this tutorial using the VS Code text editor, then after you've mastered the basics explore using the other editing modes.

### Basic Workflow

Quarto `.qmd` files contain a combination of markdown and executable code cells. Here's what it might look like in VS Code to edit and preview a `.qmd` file:



On this page

- Overview
- Basic Workflow
- Render and Preview
- YAML Options
- Markdown
- Code Cells
- External Preview
- Next Up

[Edit this page](#)  
[Report an issue](#)

Quarto Website Tutorial: VS Code

# Markdown

- Eine Markup Language (Auszeichnungssprache) zur Erzeugung formatierten Texts
- Eine HTML Alternative zur Erstellung von Webseiten etc. mithilfe einfacher Texteditoren
- Von John Gruber und Aaron Swartz 2004 mit dem Ziel hoher Lesbarkeit entwickelt

Text using Markdown syntax	Corresponding HTML produced by a Markdown processor	Text viewed in a browser
<pre>Heading *****  Sub-heading -----  # Alternative heading  ## Alternative sub-heading  Paragraphs are separated by a blank line.  Two spaces at the end of a line produce a line break.</pre>	<pre>&lt;h1&gt;Heading&lt;/h1&gt;  &lt;h2&gt;Sub-heading&lt;/h2&gt;  &lt;h1&gt;Alternative heading&lt;/h1&gt;  &lt;h2&gt;Alternative sub-heading&lt;/h2&gt;  &lt;p&gt;Paragraphs are separated by a blank line.&lt;/p&gt;  &lt;p&gt;Two spaces at the end of a line&lt;br /&gt; produce a line break.&lt;/p&gt;</pre>	<p>Heading</p> <p>Sub-heading</p> <p>Alternative heading</p> <p>Alternative sub-heading</p> <p>Paragraphs are separated by a blank line.</p> <p>Two spaces at the end of a line produce a line break.</p>
<pre>Text attributes <i>italic</i>, <b>bold</b>, 'monospace'.  Horizontal rule:  ---</pre>	<pre>&lt;p&gt;Text attributes &lt;em&gt;italic&lt;/em&gt;, &lt;strong&gt;bold&lt;/strong&gt;, &lt;code&gt;monospace&lt;/code&gt;.&lt;/p&gt;  &lt;p&gt;Horizontal rule:&lt;/p&gt;  &lt;hr /&gt;</pre>	<p>Text attributes <i>italic</i>, <b>bold</b>, <code>monospace</code> .</p> <p>Horizontal rule:</p> <hr/>

- Ein Softwarepaket zur Vereinfachung von TeX
- TeX ist ein von Donald E. Knuth ab 1977 entwickeltes Textsatzsystem mit Makrosprache
- LaTeX wurde von Leslie Lamport Anfang 1984 entwickelt
- LaTeX ist insbesondere für mathematische Berichte und Präsentationen (Beamer) nützlich

```
\footnotesize
\begin{theorem}[Datenverteilung des Allgemeinen Linearen Modells]
\justifying
\normalfont
Es sei
\begin{equation}
\upsilon = X\beta + \varepsilon \text{ mit } \varepsilon \sim N(0_n, \sigma^2 I_n)
\end{equation}
das ALM. Dann gilt
\begin{equation}
\upsilon \sim N(\mu, \sigma^2 I_n) \text{ mit } \mu := X\beta \text{ in } \mathbb{R}^n.
\end{equation}
\end{theorem}
```



## Theorem (Datenverteilung des Allgemeinen Linearen Modells)

Es sei

$$v = X\beta + \varepsilon \text{ mit } \varepsilon \sim N(0_n, \sigma^2 I_n) \quad (7)$$

das ALM. Dann gilt

$$v \sim N(\mu, \sigma^2 I_n) \text{ mit } \mu := X\beta \in \mathbb{R}^n. \quad (8)$$

- Guide
- Authoring
- Computations
- Tools
- Documents
- Presentations
- Dashboards
- Websites
- Books
- Manuscripts
- Interactivity
- Publishing
- Projects
- Advanced

## Guide

Comprehensive guide to using Quarto. If you are just starting out, you may want to explore the tutorials to learn the basics.

### Authoring

- Create content with markdown
- Markdown Basics
- Figures
- Tables
- Diagrams
- Citations & Footnotes
- Cross References
- Article Layout

### Computations

- Execute code and display its output
- Using Python
- Using R
- Using Julia
- Using Observable
- Execution Options
- Parameters

### Tools

- Use your favorite tools with Quarto
- JupyterLab
- RStudio IDE
- VS Code
- Neovim
- Text Editors
- Visual Editor

### Documents

- Generate output in many formats
- HTML
- PDF
- MS Word
- Typst
- Markdown
- All Formats

### Presentations

- Present code and technical content
- Presentation Basics
- Reveal.js (HTML)
- PowerPoint (Office)
- Beamer (PDF)

### Dashboards

- Publish data with dashboards
- Dashboard Basics
- Layout
- Data Display
- Interactivity
- Deployment

### Websites

- Create websites and blogs
- Creating a Website
- Website Navigation
- Creating a Blog
- Website Search
- Website Listings

### Books

- Create books and manuscripts
- Creating a Book
- Book Structure
- Book Crossrefs
- Customizing Output

### Manuscripts

- Write and publish notebook-first scholarly articles
- Getting Started
- Authoring Manuscripts
- Publishing Manuscripts
- Using Manuscripts

### Interactivity

- Engage readers with interactivity
- Overview
- Observable JS
- Shiny
- Widgets
- Component Layout

### Publishing

- Publishing documents and sites
- Publishing Basics
- Quarto Pub
- GitHub Pages
- Posit Connect
- Posit Cloud
- Netlify
- Confluence
- Other Services

### Projects

- Scale up your work with projects
- Project Basics
- Managing Execution
- Project Profiles
- Environment Variables
- Project Scripts
- Virtual Environments

# Quarto Beispiel

```
---
title: "Quarto Demonstration"
author: "Belinda Fleischmann"
date: today
format: pdf
---

# Überschrift zu Kapitel 1.

Hier steht der Text für Kapitel 1. Darin könnte auch eine Abbildung enthalten sein.

{width="10%"}

## Überschrift zum Unterkapitel 1.1

Hier steht der Text für Unterkapitel 1.1. Manche Worte möchte ich  fett  und manche Worte  kursiv , und Befehle in  monospace  schreiben. Mögliche Farben möchte ich mit Stichpunkten auflisten.

*  \textcolor{blue}{blau} 
*  \textcolor{green}{grün} 
*  \textcolor{red}{rot} 
*  \textcolor{gray}{grau} 

Wenn wir mathematische Ausdrücke mit Dollarzeichen umrahmen, werden sie mithilfe von  \LaTeX  formatiert. So können wir z.B. die Verteilung eines Zufallsvektors formal mit   $\epsilon \sim N(\mu, \sigma^2 I_n)$   mit   $\mu := X\beta \in \mathbb{R}^n$   aufschreiben.
```

## Quarto Demonstration

Belinda Fleischmann

2024-05-02

### Überschrift zu Kapitel 1.

Hier steht der Text für Kapitel 1. Darin könnte auch eine Abbildung enthalten sein.



### Überschrift zum Unterkapitel 1.1

Hier steht der Text für Unterkapitel 1.1. Manche Worte möchte ich **fett** und manche Worte *kursiv*. und Befehle in `monospace` schreiben. Mögliche Farben möchte ich mit Stichpunkten auflisten.

- blau
- grün
- rot
- grau

Wenn wir mathematische Ausdrücke mit Dollarzeichen umrahmen, werden sie mithilfe von LaTeX formatiert. So können wir z.B. die Verteilung eines Zufallsvektors formal mit  $v \sim \mathcal{N}(\mu, \sigma^2 I_n)$  mit  $\mu := X\beta \in \mathbb{R}^n$  aufschreiben.

Beispielbericht

Beispielpräsentation

R Tools

Quarto

**Zotero**

## Was ist ein Reference Manager?

- Reference Manager sind Literaturverwaltungsprogramme
- Reference Manager unterstützen Zitationen und das Erstellen von Literaturverzeichnissen
- Zitierstile können automatisch auf bestimmte Spezifikationen (z.B. APA) eingestellt werden
- Reference Manager dienen auch als digitale Bibliotheken
- Kommerzielle Reference Manager sind z.B. EndNote, Citavi, Mendeley und Papers
- Kostenlose/Freemium Reference Manager sind z.B. [JabRef](#) und [Zotero](#)
- Eine Integration in Quarto erlaubt z.B. der Export der eigenen Library in das [BibTex](#) Format.

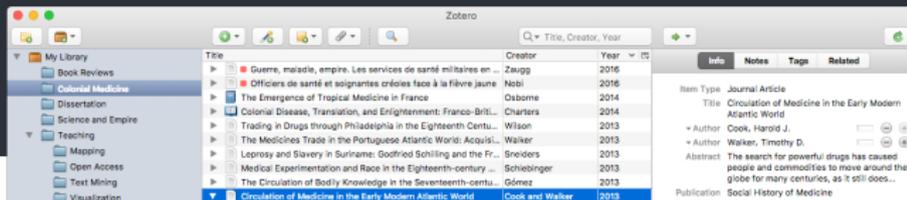
## Your personal research assistant

Zotero is a free, easy-to-use tool to help you  
collect, organize, annotate, cite, and share research.

Download

Available for Mac, Windows, Linux, and iOS

Just need to create a quick bibliography? Try [ZoteroBib](#).



[Zotero Website](#)

[Zotero Documentation](#)