Datenvisualisierung mit ggplot2\_Teil2

Sebastian Kuhn

4 9 2021

# Datenvisualisierung mit ggplot2 - Videocasts

## Inhalt

* [Datenvisualisierung mit ggplot2 - Teil 1: Grundlagen](#basics)
  + [Die “Grammar of Graphics”](#grammar)
  + [Ein erster ggplot-Befehl](#Befehl)
  + [Literatur, Blogs und schnelle Hilfe](#Hilfe)
* [Datenvisualisierung mit ggplot2 - Teil 2: Das erste Scatterplot](#scatter)
  + [Und los geht’s!](#los)
  + [ggplot-Befehle sinnvoll aufbauen](#Aufbau)

## Datenvisualisierung mit ggplot2 - Teil 1: Grundlagen

Herzlich Willkommen zum ersten Teil einer kleinen Video-Tutorial-Reihe zum Thema “Datenvisualisierung mit ggplot2”. In diesem ersten Teil der Reihe werden wir uns die grundlegende Funktionsweise des Pakets ggplot anschauen und dafür erstmal auf einer etwas theoretischeren Ebene stehen bleiben - aber keine Angst: die Praxisbeispiele kommen noch früh genug in den nächsten Videos! Vorher glaube ich allerdings, dass es erstmal ganz sinnvoll ist, sich einen Überblick über die Grundprinzipien und die Ideen zu verschaffen, die mit ggplot verbunden sind - und genau das soll das Thema dieses Videos sein.

Wenn wir damit fertig sind, werde ich Ihnen in den kommenden Videos anhand einiger Beispiele zeigen, wie man Diagramme in ggplot Schritt für Schritt aufbaut. Wir starten dabei mit einem Streudiagramm und werden uns später weitere Darstellungsformen und auch komplexere Grafiken anschauen. Am Ende werden Sie - zumindest hoffe ich das - die Grundprinzipien einer effektiven Datenvisualisierung verstehen, und auch wissen, wie man mit ggplot eine ziemlich breite Palette an Diagrammen erstellen kann.

Vorab noch ein kurzer Disclaimer: Ich gehe in dieser Einführung davon aus, dass Sie die benötigte Software - also vor allem R und R-Studio - schon installiert haben und dass Sie auch schon Grundkenntnisse in R haben. Zudem ist es mir wichtig, hier zu sagen, dass ich auch mit einer Videoreihe keinen umfassenden Überblick über alle Funktionen von ggplot geben kann - vielmehr müssen wir hier zwangsläufig an der Oberfläche bleiben und auch die Beispiele, die ich Ihnen zeigen werde sind keinesfalls als “Kochrezepte” oder als strikte Regelwerke zu verstehen. Wie fast immer bei der Arbeit mit R gibt es auch bei ggplot häufig mehr als einen Weg zum Ziel - und gerade bei der Visualisierung von Daten stehen Ihnen zahllose Möglichkeiten zur Verfügung die zum Teil auch mit Geschmack zu tun haben. Das bedeutet allerdings auch, dass ggplot2 natürlich sehr viel mehr kann als Ihnen im Rahmen dieser Reihe hier zeigen kann.

Das Ziel dieser Tutorialreihe besteht im Grunde darin, dass Sie einen schnellen Eindruck von der Funktionalität von ggplot bekommen und schnell selbst damit loslegen können, Plots damit zu erstellen. Sie sollten am Ende der Reihe dazu in der Lage sein, einen in ggplot2 geschriebenen Code im Prinzip zu verstehen, wenn Sie ihn sehen und Sie sollten einen Blick dafür bekommen, wie die “Grammatik” von ggplot funktioniert, wie die verschiedenen Ebenen, Formen und Daten zusammengesetzt werden und wie Sie Abbildungen in ggplot selbst Schritt für Schritt aufbauen können.

Noch ein Hinweis an der Stelle: Haben Sie keinen zu großen Respekt vor kompliziert aussehenden Codes - seien Sie sich versichert, dass die wenigsten Menschen, die praktisch täglich mit ggplot arbeiten sämtliche Teile des Codes auswendig und fehlerfrei herunterschreiben können. Natürlich wird bei der Erstellung von schicken Grafiken mit ggplot andauernd nebenher gegoogelt und auf Cheat-Sheets gespickelt - und das ist auch okay so. Ganz im Gegenteil: Im Netz findet sich eine sehr aktive Community, die zahlreiche Tipps und Tricks zu ggplot teilt und auch häufig die Codes und Daten für unglaublich tolle Darstellungen zur Verfügung stellt. Ich werde Ihnen da am Ende des Tutorials noch zwei, drei Hinweise geben. Nutzen Sie diese Möglichkeiten und seien Sie kreativ - das ist letztlich auch Sinn und Zweck der Vernstaltung von ggplot2.

### Die “Grammar of Graphics”

So, jetzt kommen wir aber endlich zum Punkt und schauen uns an, was es mit ggplot2 auf sich hat. ggplot gehört zu der tidyverse-Familie, die eine Reihe sehr nützlicher Pakete aus dem Umfeld von R-Studio-Entwicklern beinhaltet und das wichtige ist - diese sind und werden auch gut gepflegt. Ursprünglich wurde ggplot von Hadley Wickham im Jahr 2005 entwickelt. Wickham hat sich dabei von einem Konzept namens “Grammar of Graphics” inspirieren lassen, das Ende der 1990er Jahre von Leland Wilkinson entwickelt wurde. Das zeigt sich im Übrigen auch an dem Namen des Pakets - die beiden “g” in ggplot eben für “grammar” und “grpahics” stehen.

Ich kann und will an der Stelle nicht zu sehr ins Detail gehen, aber ich glaube es macht durchaus Sinn, sich den Grundgedanken der “Grammar of Graphics” einmal anzuschauen. Die Idee von Wilkinson besteht darin, eine ‘Grammatik’ für Grafiken zu entwickeln - das sagt ja auch schon der Name. Vielleicht fragen Sie sich jetzt, was hat denn “Grammatik” mit Grafik zu tun?. Wenn Sie sich die Sprache vorstellen, dann spielt da die Grammatik eine ganz wichtige Rolle bei der systematischen Beschreibung von Sprache und Grammatik stellt im Prinzip ein Regelwerk zur Verwendung der Sprache zur Verfügung. Beherrscht man dieses Regelwerk, dann ist auf dieser Basis eine unzählige Anzahl an (mehr oder weniger) sinnvollen sprachlichen Aussagen möglich. Analog dazu soll die “Grammar of Graphics” eine Art Sammlung von Konzepten und Regeln darstellen, aus denen wir im Prinzip alle denkbaren und sinnvollen Grafiken und Datenvisualisierungen erstellen können. In seinem - übrigens sehr lesenswerten - Aufsatz zu den Grundlagen von ggplot schreibt Wickham selbst, dass diese Grammatik die Grundlage zur Erstellung von graphischen “Gedichten” darstellt. Da wird schon deutlich, dass neben diesem systematischen Aspekt der Grammatik auch ein künstlerischer Aspekt eine Rolle spielt, aber bevor wir uns diesem widmen können müssen wir uns erstmal die Bestandteile der Grammatik genauer anschauen um dazu in der Lage zu sein, systematisch Grafiken mit ggplot2 erstellen zu können.

Schauen wir uns nun die Bestandteile der Grammatik einmal näher an. In der Pyramide links sehen sie die wichtigsten Elemente der “Grammar of Graphics” und das Besondere daran ist, dass Sie mit Hilfe dieser Komponenten so ziemlich jede erdenkliche statistische Grafik beschreiben - und mit ggplot eben auch erstellen - können.

#### Data

Ganz unten finden Sie da die Daten - und klar, ohne Daten gäbe es nichts, was Sie irgendwie visualisieren könnten. Daten helfen Ihnen Daten aber auch nicht viel - vielmehr müssen Sie irgendwie festlegen, welche Variablen aus Ihren Daten in welcher Form abgebildet werden sollen. Und hiermit ist dann auch das “Aesthetics”-Element der “Grammar of Graphics” angesprochen.

#### Aesthetics

Mit ästhetischen-Funktionen verlinken Sie letztlich in ggplot2 Ihre Variable mit bestimmten ästhetischen Darstellungen in der Grafik - beispielsweise legen Sie hier also fest, welche Variablen auf der x- oder y-Achse dargestellt werden sollen oder ob Ihre Variablen mit bestimmten Farben in der Darstellung verbunden werden. Weil es hier um die logische Verbindungen zwischen den Daten und den sichtbaren Elementen in Ihrer Darstellung geht, wird an der Stelle häufig vom sogenannten “aesthetic mapping” gesprochen.

#### Scale

Sobald dann klar ist, welche Daten für eine Visualisierung verwendet werden sollen, muss auch die Skalierung festgelegt werden. Wenn wir hier von Skalen oder Skalierung sprechen meinen wir beispielsweise die Wertebereiche der x- und y-Achsen. Es macht zum Beispiel keinen Sinn, eine Zeitachse von 1800 bis 2020 darzustellen, wenn Sie nur Daten aus den Jahren 2010 bis 2020 verwenden. Das ist klar. Mit der Skalierung sind aber auch Farb- oder Größenskalen und zahlreiche andere Eigenschaften und Parameter von ggplot-Objekten gemeint. Das wird vermutlich klarer, wenn wir uns die Beispiele in den nächsten Videos anschauen.

#### Geometric Objects

Sobald feststeht, welche Daten in einem Plot dargestellt werden sollen muss wir natürlich auch festlegen, in welcher Form die Daten visualisiert werden - ob Sie zum Beispiel Punktwolke visualisiert haben möchten, Boxplots oder Balkendiagramme - und wenn wir von derlei Visualisierungen sprechen (im ggplot2-Kontext), dann ist hier die Rede von “geometrischen Objekten”. Für praktisch alle gängigen Formen von Datenvisualisierungen existieren in ggplot mittlerweile eigene geometrische Funktionen, zum Beispiel erstellt die Funktion geom\_point() Streudiagramme, geom\_bar() erstellt Balkenplots und geom\_boxplot() erstellt Boxplots.

#### Statistics

Als nächstes Element der “Grammar of Graphics” kommen wir zu statistischen Transformationen oder zu den “Statistics”. Was ist darunter zu verstehen? Nicht immer wollen Sie den exakten Wert Ihrer Variablen in einer Abbildung darstellen, vielmehr sind Sie vielleicht an Mittelwerten, Summen oder Auszählungen interessiert oder wollen z.B. den Logarithmus von Werten verwenden. Daher wird in der “Grammar of Graphics” und in ggplot2 die Möglichkeit einer statistischen Transformation vor der eigentlichen Darstellung offen gelassen. Viele Darstellungen wie z.B. Boxplots oder Histogramme setzen solche statistischen Transformationen ja im Prinzip von vorneherein voraus - bei anderen sind sie dagegen eher optional. Wenn Sie keine statistischen Transformation vornehmen, dann ist im Kontext von ggplot2 immer mal wieder die Rede von der “transformation identity”. Dann werden die Daten nicht verändert, sondern direkt aus dem Datensatz an die jeweilige Ästhetik weitergegeben.

#### Facets

Das nächste Element in der Pyramide sind die “Facets” - also auf Deutsch so etwas wie Facettenspezifikationen. Konkret ist hier gemeint, dass die ausgewählten Daten vor der eigentlichen Visualisierung noch in verschiedene Teil-Datensätze aufgesplittet werden können. Sie könnten z.B. Ihre Daten nach Ländern oder Jahren separieren und auch in separaten Darstellungen plotten. In diesem Fall verwenden wir dann eine Facettenspezifikation, die zum Beispiel für jedes Land erst einen eigenen Teildatensatz erstellt und erst dann die eigentliche Darstellung.

#### Coordinate System

Zuletzt ist für die Erstellung einer Grafik immer die Festlegung eines bestimmten Koordinatensystems notwendig, das auch zu den Daten und den geometrischen Objekten passt. Am häufigsten wird hier das sogenannte “kartesische” Koordinatensystem verwendet - für viele Darstellungen - wie zum Beispiel für Streudiagramme - passt das meistens, für andere Darstellungen - wie für Kuchendiagramme - bieten sich aber andere Systeme wie zum Beispiel das polare Koordinatensystem an.

#### Bahnhof

So, vielleicht verstehen Sie jetzt nur noch Bahnhof fragen sich, warum das alles so kompliziert sein muss - und ich gebe zu, dass das alles erstmal ziemlich viel und ziemloich trocken erscheint, ich glaube aber, dass Sie sehr davon profiteren werden, wenn sie zumindest ein Gefühl für die Grundidee von ggplot haben, bevor Sie selbst richtig loslegen.

Und zur Beruhigung: ggplot wird Ihnen viel Arbeit abnehmen, da die Entwickler sich hier viele Gedanken über intelligente Voreinstellungen und default-Werte gemacht haben und Sie letztlich einfache Grafiken schon durch die Angabe der Daten, der “Aesthetics” und der geometrischen Objekte erstellen können ohne dass Sie all die anderen Elemente, die wir gerade angesprochen haben, händisch festlegen oder verändern müssten. Wenn Sie keine weiteren Angaben - also außer der Daten, der “Aesthetics” und der geometrischen Objekte - mehr machen, dann verwendet ggplot grundsätzlich eine Reihe von sehr sorgfältig ausgewählten default-Werte, die in aller Regel zu sehr vernünftigen und vorzeigbaren Ergebnissen führen. Wenn Sie natürlich eine Darstellung für eine Publikation erstellen wollen oder tiefere Einsichten in Ihre Daten haben wollen, dass werden Sie in der Regel noch einiges anpassen müssen - für einen schnellen Einblick in die Daten sind die Voreinstellungen aber - in den meisten Fällen - völlig ausreichend.

### Ein erster ggplot-Befehl

So, jetzt wollen wir uns aber mal anschauen, wie diese Grundsätzte der “Grammar of Graphics” in ggplot umgesetzt sind. Dafür schauen wir uns jetzt einfach mal einen einfachen Befehl in ggplot schematisch an. Hier auf der Folie finden Sie diesen Befehl.

Was hier passiert ist im Grunde das Folgende: Mit der Funktion ggplot() teilen Sie R im Prinzip einfach mit, dass Sie jetzt gerne ggplot verwenden würden. Diese Funktion erwartet als Argument in aller Regel immer die Angabe des Datensatzes (und zwar des Datensatzes in dem sich die Variablen befinden, die eben dargestellt werden sollen). Sie übergeben diesen Datensatz im Argument data =, wobei diese Angabe im Prinzip optional ist. Die Funktion ggplot() erwartet grundsätzlich als erstes einen Datensatz, daher können Sie die Angabe von “data =” im Prinzip auch einfach weglassen. In dem vorliegenden Fall übergeben Sie der Funktion ggplot() also einen fiktiven Datensatz namens “df” und legen als nächstes in der Aesthetics-Funktion (also a-e-s) fest, welche Variablen Sie nun konkret verwenden wollen und stellen das sogenannte “Mapping” zwischen den Daten und Ihrer Abbildung her. In diesem Fall wollen Sie also die fiktive Variable x auf der x-Achse darstellen und die fiktive Variable y auf der y-Achse. Damit haben Sie die Daten und die Ästhetik des Plots festgelegt. Nun muss ggplot nur noch wissen, welche Darstellungsform verwendet werden soll. Daher fügen Sie nun mit einem + eine der zahlreichen geom\_-Funktionen hinzu - in diesem Fall sollen Punkte geplottet werden - es soll also ein Streudiagramm erstellt werden.

Manchmal macht es Sinn, sich einen ggplot-Befehl einmal ausgesprochen vorzustellen - und das würde in der Variante wie folgt klingen: “Hey ggplot, nimm doch bitte den Datensatz gf, stelle mir die Werte der Variable x auf der x-Achse dar und die Werte der Variable y auf der y-Achse. Und zeichne dann bitte für jede Wertkombination einen Punkt.”

Und das war’s dann auch schon - alle anderen Entscheidungen - beispielsweise über die Art des Koordniatensystems oder die Skalierung - würde ggplot in diesem Fall selbst treffen (da Sie es nicht getan haben) - und in aller Regel würde das dann auch zu einem vorziegbaren Ergebnis führen.

### Literatur, Blogs und schnelle Hilfe

So, und weil das alles in der Theorie nicht so richtig viel Spaß macht, freue ich mich schon auf die kommenden Videos in denen ich Ihnen zeigen werde, wie ggplot in der Praxis funktioniert. Vorher möchte ich Ihnen zum Ende des Videos aber noch ein paar Tipps zur Literatur und Hilfestellungen rund um ggpplot an die Hand geben. Ich habe ja schon am Anfang des Videos gesagt, dass Sie keinen allzu großen Respekt vor kompliziert aussehenden Codes haben sollten, sondern sich mit Offenheit und mit Kreativität auf ggplot einlassen sollten - und aus meiner Erfahrung heraus ist es hierzu sehr hilfreich, sich einfach mal im Netz anzuschauen, was andere Menschen so alles mit ggplot machen und dann auch gegebenenfalls besonders gelungene oder schöne Grafiken einfach einmal selbst zu replizieren. Glücklicherweise gibt es dazu mittlerweile im Netz einen Haufen toller Blogs und ich habe Ihnen hier nur einmal eine ganz kleine Auswahl mitgebracht.

Sehr ans Herz legen will ich Ihnen beispielsweise die [R-graph-gallery](https://www.r-graph-gallery.com/), die eine Vielzahl an möglichen ggplots beinhaltet und auch gleich den Code dazu bereitstellt - und auch beispielsweise den Blog von [Cédric Scherer](https://www.cedricscherer.com/), der wirklich sehr schöne und informative Grafiken erstellt - wie zum Beispiel auch die, die Sie hier auf der Folie sehen können - und der auch zahlreiche Tutorials zu ggplot2 auf seinen Seiten bereit hält - da lohnt sich ein Blick auf jeden Fall.

Dann finden Sie hier auf der Folie auch zwei Literaturempfehlungen - die beiden Bücher von Kiran Healy und von Claus Wilke halte ich wirklich für uneingeschränkt empfehlenswert und auch hier ist das schöne, dass beide Autoren ihren Code auf ihren Homepages bzw. auf den github-Seiten auch frei zur Verfügung stellen. Das heißt, Sie können im Grunde alle Abbildungen, die Sie in den Büchern finden auch mit mehr oder weniger großem Aufwand selbst auch nachbauen.

Zuletzt will ich Sie noch auf die umfangreiche Dokumentation des Pakets auf den [tidyverse](https://ggplot2.tidyverse.org/)-Seiten hinweisen - hier finden Sie in der Regel schnell Informationen zu allen Funktionen von ggplot2 - und übrigens auch einen sehr hilfreichen Spickzettel mit den wichtigsten Befehlen und Funktionen, den Sie einfach ausdrucken und auf Ihren Schreibtisch legen können. Wenn auch das nicht mehr hilft, dann lege ich Ihnen zum Schluss noch die Plattform [stackoverflow](https://stackoverflow.com/) ans Herz - hier finden Sie insbesondere dann Rat wenn Sie ganz besonders knifflige Fragen oder Probleme haben - da lohnt sich dann ein Blick wenn Sie nicht mehr weiter kommen.

Und nun wünsche ich Ihnen schonmal viel Spaß beim stöbern und freue mich auf die kommenden Videos!

## Datenvisualisierung mit ggplot2 - Teil 2: Das erste Scatterplot

Herzlich Willkommen zum zweiten Teil der Video-Tutorial-Reige zu ggplot2. Nachdem ich im ersten Video einiges über die theoretischen Grundlagen von ggplot2 und über die Grundidee der “Grammar of Graphics” erzählt habe, wird es jetzt Zeit, die Theorie zu verlassen und zur Praxis zu kommen.

### Die Datenbasis

Bevor wir aber unser erstes ggplot gemeinsam erstellen, will ich kurz etwas zur Datenbasis sagen, die ich hier verwende. Sie sehen oben hier im Environment, dass bereits ein Datensatz geladen ist - und zwar handelt es sich dabei um einen Auszug aus dem *Quality of Government*-Datensatz des [Quality of Government Institute](https://www.gu.se/en/quality-government) der Universität Göteborg. Der Fokus dieses Datensatzes liegt - wie es der Name erahnen lässt - auf *Good Governance* und beinhaltet Mikro- und Makrodaten aus unterschiedlichen Quellen, die für über 190 Länder aufbereitet werden. Die Daten sind dabei thematisch - beispielsweise nach den Bereichen Bildung, Gesundheit, Migration und Umwelt - gegliedert und der Datensatz liegt in mehreren Varianten vor - so beinhaltet der [Standard-Datensatz](https://www.gu.se/en/quality-government/qog-data/data-downloads/standard-dataset), den wir hier auch nutzen, beispielsweise über 2.000 Variablen wohingegen der [Basic-Datasatz](https://www.gu.se/en/quality-government/qog-data/data-downloads/basic-dataset) eine kompaktere Version mit nur etwa 300 Variablen darstellt. Darüber hinaus liegen viele Daten auch in einem [Zeitreihendatensatz](https://www.gu.se/en/quality-government/qog-data/data-downloads/standard-dataset) vor. Für dieses Tutorial habe ich einen Auszug aus dem Standard-Datensatz sowie aus dem Zeitreihen-Datensatz erstellt, später werden wir auch einen Auszug aus dem Zeitreihen-Datensatz verwenden, und diese Auszüge enthalten einige Variablen weniger, so dass der Datensatz nicht mehr ganz so groß ist.

Wie bereits erwähnt, habe ich diesen Auszug aus dem Datensatz in der Session bereits geladen, im Skript finden Sie aber hier unter “Import des QOG Standard-Datensatzes” und “Import des QOG Zeitreihen-Datensatzes” Beispiele zum Laden der vollständigen Datensätze über die Original-URL des Quality of Government Institute. Bevor wird die Datensätze importieren, laden wir via library(tidyverse) alle Pakete des tidyverse-Universums - darunter neben ggplot2 auch das Paket readr, welches auch die Funktion read\_csv() beinhaltet, die wir hier verwenden.

## Wichtige Pakete laden und Datensatz importieren  
library(tidyverse) # lädt alle Pakete des "tidyverse" - unter anderem ggplot2 und readr  
  
# Import des QOG Standard\_Datensatzes  
  
qog <- readr::read\_csv("~/Dokumente/R Projekte/ggplot\_FUH\_Videocast/Data/qog\_sample.csv")  
#qog <- readr::read\_csv("/home/seba/Dokumente/R Projekte/ggplot\_tut/Data/qog\_std\_cs\_jan21.csv")  
#qog <- readr::read\_csv("https://www.qogdata.pol.gu.se/data/qog\_std\_cs\_jan21.csv") #Standard-Datensatz  
  
# Import des QOG Zeitreihen-Datensatz  
  
qog\_ts <- readr::read\_csv("~/Dokumente/R Projekte/ggplot\_FUH\_Videocast/Data/qog\_ts\_sample.csv")  
# qog\_ts <- readr::read\_csv("/home/seba/Dokumente/R Projekte/ggplot\_tut/Data/qog\_std\_ts\_jan21.csv")  
#qog\_ts <- readr::read\_csv("https://www.qogdata.pol.gu.se/data/qog\_std\_ts\_jan21.csv") #Zeitreihen-Datensatz

### Und los geht’s!

#### Empfehlung

So, und damit geht’s los. Bevor wir anfangen möchte ich Ihnen empfehlen, alle Übungen die wir hier machen auch tatsächlich manuell durchzuführen, indem Sie den Code wirklich selbst eintippen (statt ihn zu kopieren und einzufügen). Jetzt werden Sie sich vielleicht fragen: “Code selbst eintippen?” Das klingt in Zeiten des “Copy und Paste” vielleicht etwas antiquiert, ist aber bei weitem der effektivste Weg, um sich an die Logik von ggplot zu gewöhnen und ein Gefühl für die Syntax von R zu bekommen. Nur wenn Sie selbst Fehler machen - und ich bin mir sicher: das werden Sie unweigerlich tun - werden Sie auch schnell in der Lage sein, die eigenen Fehler zu diagnostizieren und daraus zu lernen.

#### Das erste Streudiagramm

Wenn wir jetzt zum ersten Streudiagramm kommen, dann nehmen wir einfach mal an, wir interessieren uns für den Zusammenhang zwischen der Höhe staatlicher Gesundheitsausgaben und der Lebenserwartung in allen Ländern, für die wir eben Daten haben. Beide Variablen finden sich im Quality of Government-Datensatz - und zwar unter der Bezeichnung wdi\_dgovhexp - für die öffentlichen Gesundheitsausgaben - und unter der Bezeichnung wdi\_lifexp - für die Lebenserwartung. Das Präfix “wdi” deutet übrigens auf die ursprüngliche Datenquelle hin - beide Variablen stammen aus den “World Development Indicators”-Datensatz der Welt Bank.

## Variablen:  
# wdi\_dgovhexp: Öffentliche Ausgaben für Gesundheit aus inländischen Quellen als Anteil der Wirtschaft, gemessen am BIP.  
# wdi\_lifexp: Die Lebenserwartung bei der Geburt gibt die Anzahl der Jahre an, die ein Neugeborenes leben würde, wenn die zum Zeitpunkt seiner Geburt vorherrschenden Sterblichkeitsmuster zum Zeitpunkt seiner Geburt sein ganzes Leben lang gleich bleiben würden.

So, jetzt wollen wir aber wirklich loslegen mit ggplot und dazu teilen wir ggplot in der Funktion ggplot() erstmal mit, in welchem Datensatz sich die Variablen befinden, die uns interessieren. Im ersten Video hatte ich ja bereits gesagt, dass die Funktion ggplot() grundsätzlich als erstes Argument Daten erwartet, in diesem Fall also die Daten “qog”. Wenn wir diesen Befehl ausführen passiert erstmal…

## Daten übergeben  
ggplot(data = qog)



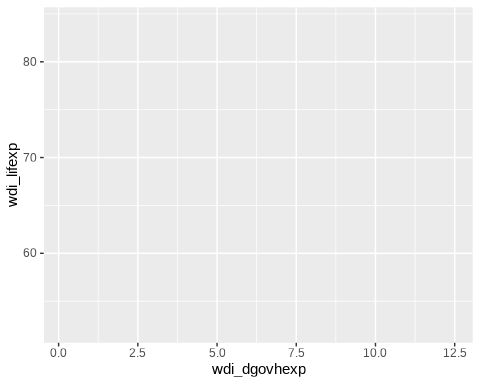
# ggplot(qog) # Führt zum selben Ergebnis

…nichts. Naja, das ist nicht ganz richtig - wie Sie gesehen haben hat sich hier im Plot-Fenster das Fenster “eingegraut”, d.h. Sie haben eine große graue Fläche erhalten - vielmehr ist aber nicht passiert. Warum?

Naja, im Prinzip kennt ggplot ja jetzt nur unsere Daten, weiß aber noch nicht, welche Variablen wir auf welche Weise darstellen wollen. Bevor wir weiter machen nochmal kurz der Hinweis, dass die Angabe von “data=” in diesem Fall optional ist - ich habe ja gesagt: die ´Funktion ggplot() geht grundsätzlich davon aus, dass als erstes Argument ein Datensatz übergeben wird und dementsprechend würde auch der Befehl ohne die Angabe “data=” zum selben Ergebnis führen.

Wir müssen ggplot also als nächstes mitteilen, welche Variablen in den Daten durch welche visuelle Elemente im Diagramm dargestellt werden sollen. Diese Zuordnung erfolgt - wie auch im ersten Video schon erläutert - über die “Aesthetics-” bzw. aes()-Funktion, die sich in aller Regel sofort an die Angabe des Datensatzes anschließt - und das sieht dann in etwa so aus:

# Das "Mapping-Argument" - die Zuordnung von Variablen zu Aesthetics  
ggplot(qog, aes(x = wdi\_dgovhexp, y = wdi\_lifexp))



Jetzt haben wir der ggplot()-Funktion also zwei Argumente gegeben: Erstens sagen wir ggplot welchen Datensatz es verwendet werden soll und wo die Variablen zu finden sind, die im Zusammenhang des Plots verwendet werden sollen. Zwar können in einer ggplot-Darstellung durchaus auch zwei oder mehr Datensätze verwendet werden, in aller Regel finden sich aber alle Variablen in einem Datensatz.

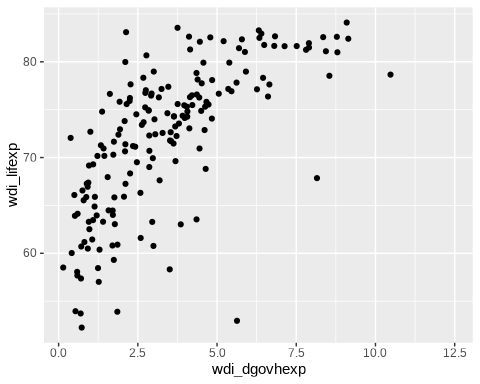
Als nächstes folgt das sogenannte “Mapping-Argument”, das innerhalb der aes()-Funktion angegeben wird. Die Argumente, die wir der Funktion übergeben, sind im Prinzip nichts anderes als Definitionen, die ggplot später verwenden wird. Im vorliegenden Fall könnte man den Codeschnipsel also folgendermaßen übersetzen:

“Nimm bitte den Datensatz”qog" zur Hand und stelle die Variable “wdi\_dgovhexp” auf der x-Achse und die Variable “wdi\_lifexp” auf der y-Achse dar". Wie wir später noch sehen werden, könnten wir Variablen hier auch zu anderen ästhetischen Elementen zuordnen - beispielsweise zu Farben, Formen oder Linientypen - erstmal bleiben wir hier aber bei der Zuordnung zu den Achsen.

Jetzt führen wir den Befehl einmal aus - und wie Sie sehen ist durch die Ausführung des Befehls etwas mehr passiert als vorhin. Die Ausgabe ist nun nicht mehr vollständig weiß oder grau, sondern R hat bereits die x- und y-Achse erstellt und beschriftet und eine grundlegende Design-Entscheidung bezüglich des Hintergrunds und einiger Hilfslinien getroffen - und das alles ohne dass Sie eine konkrete Aufforderung im ggplot-Code geschrieben hätten. Das liegt daran, dass für diese Ebenen bereits Voreinstellungen definiert wurden, die nun standardmäßig abgerufen werden.

Das wichtigste aber - nämlich die Darstellung der beiden Variablen - fehlt noch. Das liegt schlicht und ergreifend daran, dass das Mapping alleine noch nichts darüber sagt, wie die Darstellung genau erfolgen soll - ob also Punkte, Balken, Linien oder andere geometrische Objekte gezeichnet werden sollen. Wir müssen dem Plot daher noch mindestens eine weitere Ebene hinzufügen, die über die Art des Diagramms entscheidet. Konkret bedeutet das, dass wir uns für eine der knapp 50 vordefinierten geometrischen Funktion entscheiden müssen, die in ggplot enthalten sind. In diesem Fall verwenden wir geom\_point() - ohne Angabe weiterer Argumente in der Klammer. Durch die bereits vorab definierte Zuordnung der jeweiligen Variablen zu den x- und y-Achsen ist jetzt schon klar, welche x- und y-Werte in einem Streudiagramm darstellt werden sollen. Hinzugefügt wird die geom\_-Funktion über ein den “+”-Operator.

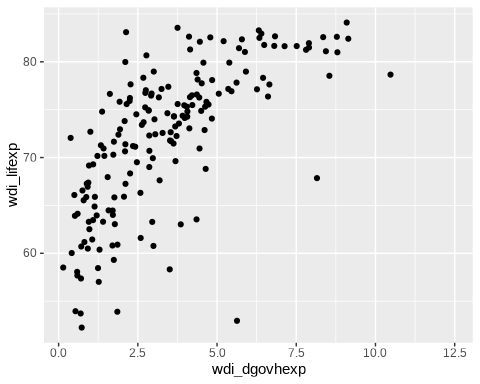
# Das erste Streudiagramm  
ggplot(qog, aes(x = wdi\_dgovhexp, y = wdi\_lifexp)) + # Neue Ebenen werden mit dem "+"-Operator hinzugefügt  
 geom\_point()



Jetzt führen wir auch diesen Befehl einmal aus - und siehe da: Schon ist das erste Streudiagramm fertig! Zugegeben, es sieht noch nicht wirklich berauschend aus und würde so vermutlich nicht in einer Studienarbeit oder in einem Journal publiziert werden, aber der erste Schritt ist schon mal getan. Bevor wir uns jetzt aber weiteren Details von ggplot2 widmen, möchte ich Sie an der Stelle noch auf einige praktische Punkte hinweisen.

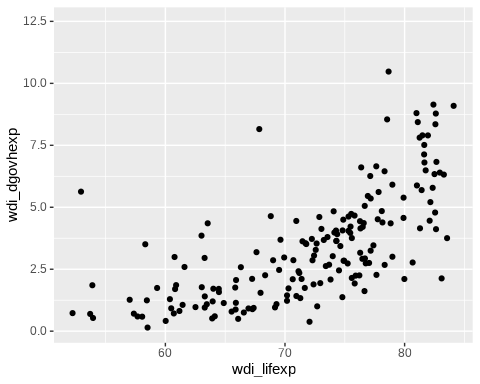
Wenn Sie in Lehrbüchern oder im Internet ggplot2-Code sehen, dann werden Sie merken, dass es sehr unterschiedliche Vorlieben bezüglich des Code-Stils gibt und dass insbesondere erfahrende R-Nutzer manchmal dazu tendieren, sehr kurzen und effizienten Code zu schreiben, der für Anfänger häufig nicht gut lesbar ist. Ein Beispiel: Mit dem folgenden Code, den ich hier markiere und einmal ausführe, kommen Sie zu dem selben Ergebnis - und das, obwohl er deutlich kürzer ist als der Code weiter oben.

# Das erste Streudiagramm - kürzere Variante  
ggplot(qog, aes(wdi\_dgovhexp, wdi\_lifexp)) + geom\_point()



Wie wir bereits oben bei der optionalen Angabe des “data=”-Arguments gesehen haben, sind einige Angaben optional, da die verwendeten Funktionen einige Voreinstellungen haben. Wenn man beispielsweise weiß, dass die Aesthetics-Funktion ohne nähere Angabe immer davon ausgeht, dass als erstes Argument eine x-Variable und als zweites Argument eine y-Variable übergeben wird, dann kann man sich die Angabe von “x=” und “y=” - wie wir sie oben gemacht hatten - auch schlicht und ergreifend sparen. Hier ist allerdings aus meiner Sicht Vorsicht geboten, denn wenn die Reihenfolge der Variablen vertauscht wird, ändert sich natürlich auch die Zuordnung der Variablen zur x- bzw. y-Achse. Das habe ich in dem Befehl hier einmal so gemacht - da ist einfach die Reihenfolge vertausch:

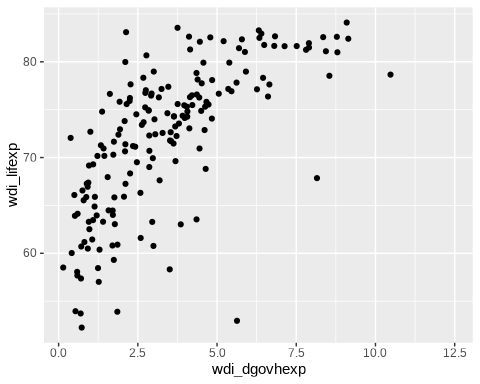
# Das erste Streudiagramm - kürzere Variante mit anderer Reihenfolge  
ggplot(qog, aes(wdi\_lifexp, wdi\_dgovhexp)) + geom\_point()



Und Sie sehen, dann werden die auch die Zuordnungen der Variablen zur x- und zur y-Achse verändert.

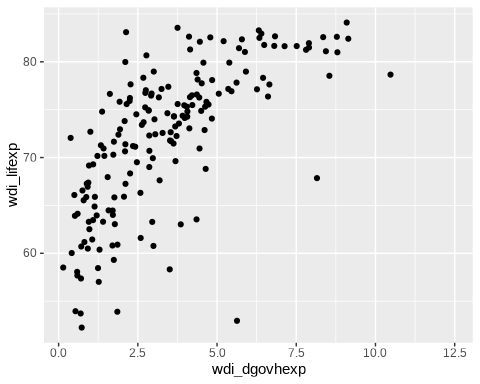
Für Anfänger bietet sich daher die explizite Angabe - auch zur besseren Lesbarkeit - durchaus an. Die explizite Angabe von “x” und “y” “schlägt” dann im Übrigen auch die Reihenfolge der Variablen - also wenn Sie mit “y=” zuerst die y-Variable angeben und dann die x-Variable (mit “x=”), dann werden Sie sehen, dass wir wieder das ursprüngliche Diagramm haben und dass die y-Variable - also in dem Fall die Lebenserwartung - auf die y-Achse geplottet wurde, obwohl es die erste Variable in der aes()-Funktion ist. Und die Gesundheitsausgaben wurden auf die x-Achse geplottet, obwohl es die zweite Variable war.

# Das erste Streudiagramm - y vor x  
ggplot(qog, aes(y = wdi\_lifexp, x = wdi\_dgovhexp)) + geom\_point()



Kein guter Stil ist die Definition neuer Ebenen in der gleichen Zeile - wie in dem Beispiel mit + geom\_point() in der selben Zeile wie der Rest des Codes. Das funktioniert zwar, aber wenn Sie neue ggplot-Ebenen auch einfach gleich in neue Zeilen schreiben, dann erhöht das erheblich die Übersicht und Lesbarkeit des Codes. Deswegen würde ich Ihnen empfehlen, sich das gleich von Anfang an anzugewöhnen.

# Das erste Streudiagramm - y vor x  
ggplot(qog, aes(y = wdi\_lifexp, x = wdi\_dgovhexp)) +   
 geom\_point() # Zur besseren Lesbarkeit und Übersicht bitte in einer neuen Zeile

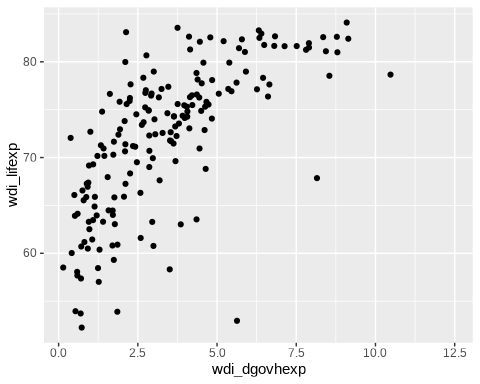


## ggplot-Befehle sinnvoll aufbauen

### ggplots als Objekte speichern

Was Sie ebenfalls in nahezu allen Lehrbüchern und Blogbeiträgen zu ggplot2 häufig sehen werden ist, dass ggplot-Nutzer ihre Grafiken meistens gar nicht direkt erstellen sondern schrittweise vorgehen und Zwischenschritte in Objekte speichern - das kommt nämlich auch der Logik der “Grammar of Graphics” im Prinzip entgegen. In unserem Fall sieht das beispielsweise so aus:

# Daten und Mapping in "p" zwischenspeichern  
p <- ggplot(qog, aes(y = wdi\_lifexp, x = wdi\_dgovhexp))  
# Darstellung als Scatterplot  
p + geom\_point()

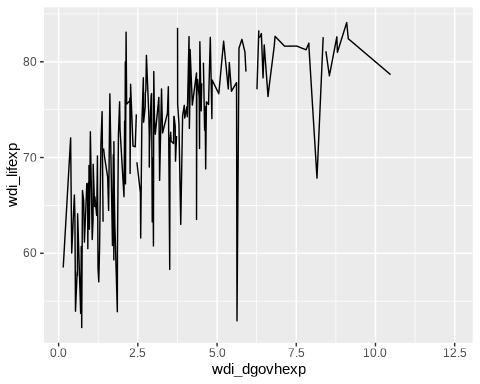


Was hier passiert, ist, dass die Informationen zum Datensatz sowie die Informationen zum Mapping der Variablen auf der x- und y-Achse in einem neuen Objekt namens “p” zwischengespeichert werden und erst im zweiten Schritt wird dann der Darstellungstyp festgelegt oder werden andere Festlegungen getroffen.

Wenn Sie diese Codeline einmal ausführen, dann passiert im Plotfenster erstmal gar nichts - was passiert, ist dass Sie im Environment ein neues Objekt namens “p” enthalten haben. Eine Darstellung im Plotfenster wird durch die Zuordnung der ggplot-Funktion zum Objekt “p” nicht ausgelöst. Wenn Sie eine Darstellung haben möchten, dann würden Sie in dem Fall einfach das Objekt “p” abrufen - und das würde in dem Fall wieder dazu führen, dass Sie (wie wir vorher in dem Beispiel schon gesehen hatten) ein mehr oder weniger leeren Plot ausgegeben bekommen in dem nur die Achsen und die Bezeichnungen angelegt sind, aber noch keine Punkte, denn: Das haben Sie ja hier in der Darstellung noch gar nicht festgelegt. Im zweiten Schritt würden Sie dann z.B. indem Sie zu dem Objekt “p” eine geom\_-Funktion hinzufügen dann (in diesem Fall ein Scatter-) Plot wirklich darstellen. Durch die Angabe “p + geom\_point()” wird Ihnen dann das gewohnte Streudiagramm ausgegeben, das wir ja schon ein paar mal gesehen haben.

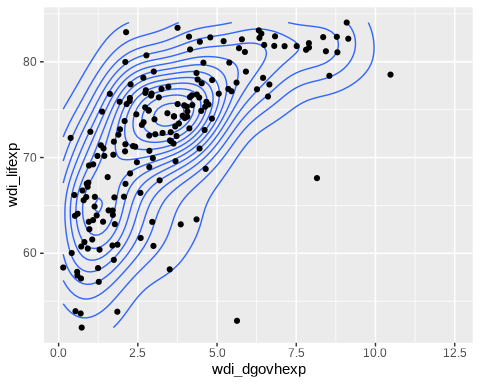
Zwei Punkte sind mir an dieser Stelle wichtig: Erstens die Erkenntnis, dass auch von ggplot erstellte Plots oder Funktionen in R als Objekte gespeichert werden können. Zweitens natürlich - und das ist der wichtige Punkt - die Frage nach dem Sinn der Übung. Meistens dient diese Arbeitsweise dazu, herumexperimentieren zu können, ohne immer wieder Code wiederholen zu müssen. Sie könnten beispielsweise auf diese Art schnell ausprobieren, ob sich nicht vielleicht ein Liniendiagramm besser zur Darstellung der Daten eignen könnte als ein Streudiagramm. In diesem Fall fügen Sie dem Objekt “p” eben nicht die Funktion geom\_point() hinzu, sondern die Funktion geom\_line() - die Ihnen ein Liniendiagramm erstellt.

p + geom\_line()



Das macht offenkundig keinen Sinn. Vielleicht wollen Sie aber mal ein Density-Plot - also ein Dichte-Diagramm - erstellen, das die Verteilung nummerischer Variable in einer bestimmten Weise abbildet - das würden Sie zum Beispiel mit diesem Code hier machen:

p + geom\_density\_2d() +   
 geom\_point()



Und auch da kann man sich jetzt darüber streiten, ob das sinnvoll war, es geht mir an der Stelle aber noch gar nicht darum zu verstehen, was hier passiert ist - wichtig ist nur: Durch diesen Aufbau des Codes (und das Zwischenspeichern von Ergebnissen in Objekten) sind Sie in der Lage, schnell und ohne viel weitere Schreibarbeit herumzuexperimentieren - zumindest solange Sie die selbe Datengrundlage und die selben Variablen verwenden. Mein Tipp: Gewöhnen Sie sich das gleich an - ich werde es in den kommenden Videos auch so machen und ich glaube es ist sinnvoll, sich das von Anfang an direkt so anzugewöhnen.

Übrigens müssen Sie das Objekt dabei natürlich nicht zwingend “p” nennen - das “p” hat sich allerdings - vermutlich als Abkürzung für “plot” - weitgehend durchgesetzt und wird Ihnen in vielen Lehrbüchern so begegnen.

Ich gebe zu, das alles mag für Anfänger etwas kompliziert erscheinen und wir haben jetzt relativ viel Zeit darauf verwendet, ein recht simples Plot zu erstellen - wichtig ist es aber gerade am Anfang das Grundkonzept von ggplot zu verstehen. Und die gute Nachricht ist nun: An diesem Konzept ändert sich auch bei komplexeren Darstellungen nichts grundsätzliches. Komplexere Darstellungen bestehen eben meistens nur aus mehr Ebenen. In den Darstellungen, die wir uns im Laufe der weiteren Videos anschauen werden, werden wir beispielsweise andere geom()-Funktionen verwenden, das Koordinatensystem, Skalen, Farben und Beschriftungen ändern - konzeptionell werden wir jedoch immer das Gleiche tun. Deswegen würde ich Ihnen gerne zum Ende des ersten Praxisvideos die zentralen konzeptuellen Schritte bei der Erstellung eines ggplots nochmal wiederholen:

Die zentralen Schritte sind - Erstens - das Sie der ggplot()-Funktion mitteilen, welche Daten Sie verwenden wollen, dass Sie dann - Zweitens - ggplot() mitteilen, welche Variablen Sie dargestellt haben wollen - und der Einfachheit halber diesen Schritt auch schon in einem Objekt zwischenspeichern. In einem dritten Schritt teilen Sie ggplot mit, welchen Darstellungstyp Sie gerne sehen wollen und in einem vierten Schritt können Sie dann weitere Zusatzfunktionen verwenden, um Skalen, Beschriftungen, Markierungen, Titel und vieles mehr hinzufügen.

# Wir beginnen mit einem möglichst "aufgeräumten" Datensatz und gehen dann wie folgt vor:  
# 1. Teilen Sie der Funktion ggplot() mit, welche Daten Sie verwenden wollen.  
# 2. Sagen Sie ggplot(), welche Variablen Sie dargestellt haben wollen. Der Einfachheit halber speichern Sie die Ergebnisse der ersten beiden Schritte in ein Objekt namens "p".  
# 3. Teilen Sie ggplot mit, welchen Darstellungstyp Sie gerne sehen wollen. Verwenden Sie dazu die vorgesehenen geom\_()-Funktionen, indem Sie diese einzeln und nacheinander zum Objekt "p" hinzufügen.  
# 4. Verwenden Sie Zusatzfunktionen, um Skalen, Beschriftungen, Markierungen, Titel und vieles mehr hinzufügen.

All’ das werden wir uns in den kommenden Videos anschauen - für heute würde ich es gerne dabei bewenden lassen. Bis dahin: viel Spaß beim Ausprobieren und bis zum nächsten Video!