4. Export und Canvas

Plotgroessen kontrollieren mit ggsave und camcorder

Dr. Paul Schmidt

```
for (pkg in c("ggplot2")) {
  if (!require(pkg, character.only = TRUE)) install.packages(pkg)
}
```

In den vorherigen Kapiteln haben wir Plots erstellt, die im RStudio-Vorschaufenster gut aussahen. Aber die Vorschau ist truegerisch: Sie passt jeden Plot an die aktuelle Fenstergroesse an, sodass Schriften, Linienstaerken und Abstaende sich bei jeder Groessenaenderung des Fensters mitaendern. Ein Plot, der in einem breiten Fenster ausgewogen wirkt, kann in einem schmalen Fenster uebergrossen Text haben - oder umgekehrt. Fuer reproduzierbare, publikationsreife Ausgaben muss man in einer festen Groesse und Aufloesung exportieren.

Dieses Kapitel behandelt zwei Werkzeuge dafuer: `ggsave()` fuer den direkten Dateiexport und das {camcorder}-Paket fuer einen WYSIWYG-Vorschau-Workflow. Beide loesen dasselbe grundlegende Problem - das **Canvas-Problem**.

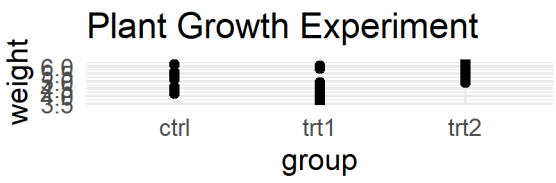
Das Canvas-Problem

Um zu verstehen, warum Exportabmessungen wichtig sind, betrachten wir folgende Demonstration. Wir erstellen einen einfachen `PlantGrowth`-Plot und rendern ihn bei zwei sehr unterschiedlichen Canvas-Groessen:

```
p <- ggplot(PlantGrowth, aes(x = group, y = weight)) +
  geom_point() +
  labs(title = "Plant Growth Experiment") +
  theme_minimal()
```

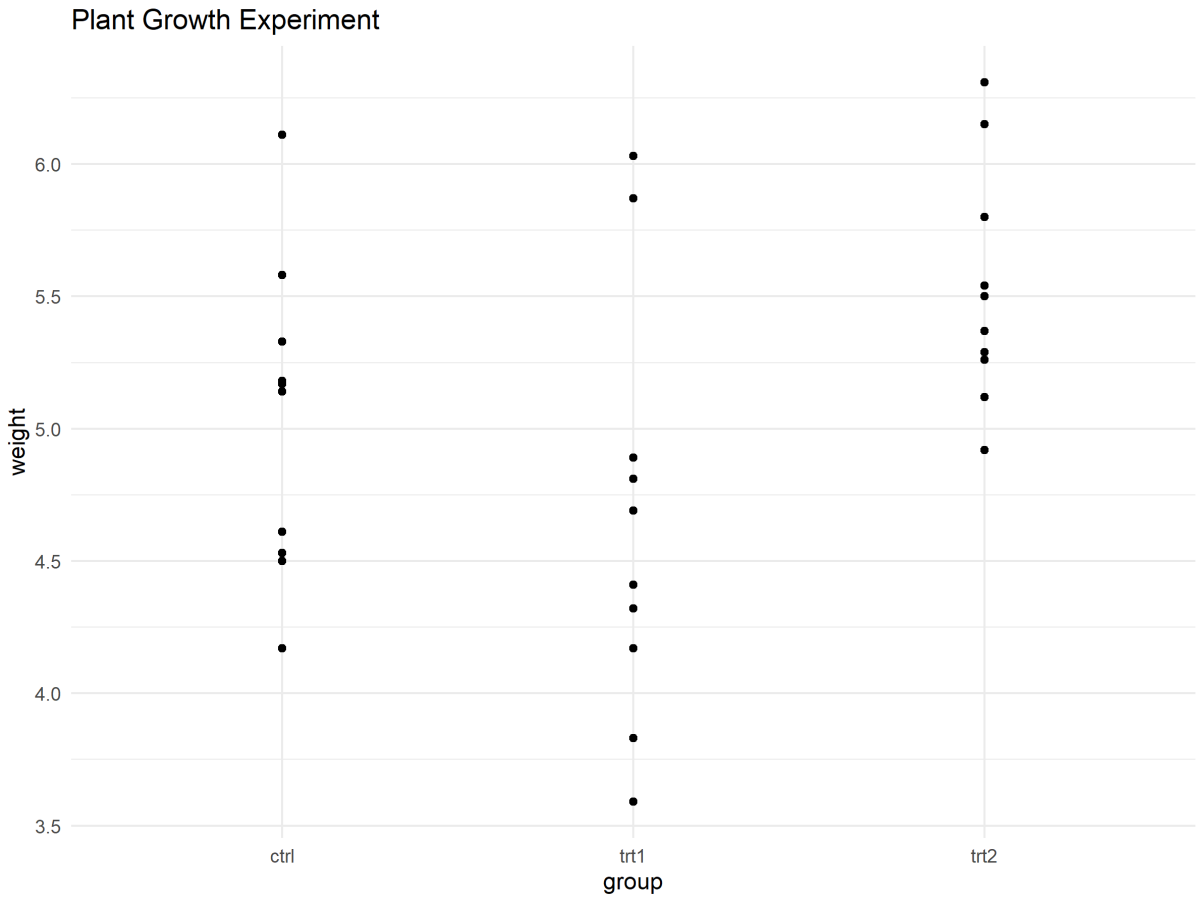
Bei einem sehr kleinen Canvas (3 x 1 Zoll) dominieren Textelemente den Plot und die Daten sind kaum sichtbar:

```
p
```



Bei einem groesseren Canvas (8 x 6 Zoll) hat derselbe Plot viel Platz, aber der Text erscheint proportional winzig:

```
p
```



Der Code ist identisch - nur die Canvas-Groesse hat sich geaendert. Das ist die zentrale Erkenntnis: **Text- und Geom-Groessen in ggplot2 sind absolut (gemessen in Punkten oder Millimetern), waehrend das Canvas relativ ist.** Ein 12pt-Titel nimmt denselben physischen Platz ein, egal ob der Plot 3 Zoll oder 8 Zoll breit ist. Auf einem kleinen Canvas fuellt 12pt-Text den groessten Teil der Flaeche; auf einem grossen Canvas wirkt er wie eine Fussnote.

Das bedeutet, dass die “richtige” Schriftgroesse, Linienstaerke und Punktgroesse alle von den Exportabmessungen abhaengen. Ein Plot, der fuer eine ganzseitige Journal-Abbildung (7 x 5 Zoll) gestaltet wurde, sieht in einer Praesentationsfolie (10 x 5.6 Zoll) falsch aus - und beides sieht im RStudio-Vorschaufenster falsch aus, das keine der beiden Dimensionen hat.

> ! Die Vorschau ist nicht die Ausgabe
>
> Das RStudio/Positron Plots-Panel skaliert jeden Plot auf die aktuelle Fenstergroesse. Das ist praktisch zum Explorieren, aber irrefuehrend fuer die finale Ausgabe. Man sollte immer die exportierte Datei (PNG, PDF etc.) pruefen, nicht die Vorschau.

ggsave

Die `ggsave()`-Funktion exportiert einen ggplot in eine Datei mit voller Kontrolle ueber Abmessungen und Aufloesung:

```
ggsave(
  filename = "my_plot.png",
  plot = p,
```

```
  width = 6,
  height = 4,
  units = "in",
  dpi = 300
)
```

Die wichtigsten Argumente sind:

- **`filename`**: Ausgabepfad. Die Dateiendung bestimmt das Format (`.png`, `.pdf`, `.svg`, `.jpg` etc.)
- **`plot`**: Das ggplot-Objekt. Wenn weggelassen, speichert `ggsave()` den zuletzt angezeigten Plot
- **`width`** und **`height`**: Canvas-Abmessungen
- **`units`**: `"in"` (Zoll), `"cm"`, `"mm"` oder `"px"`
- **`dpi`**: Aufloesung in Dots per Inch. 300 ist Standard fuer Druck, 150 fuer Bildschirm, 72 fuer schnelle Vorschauen
- **`scale`**: Multiplikator fuer die Canvas-Groesse. `scale = 2` verdoppelt alle Dimensionen, was die relative Groesse von Text und Punkten effektiv halbiert

> Gaengige Exportgroessen
>
> Einige typische Dimensionskombinationen:
>
> | Anwendungsfall | Breite | Hoehe | DPI |
> |---|---|---|---|
> | Journal-Abbildung (einspaltig) | 3.5 in | 3 in | 300 |
> | Journal-Abbildung (volle Breite) | 7 in | 5 in | 300 |
> | Praesentationsfolie (16:9) | 10 in | 5.6 in | 150 |
> | Poster-Panel | 8 in | 6 in | 300 |
> | Web/Blog | 8 in | 5 in | 150 |

Ein typischer Workflow ist iterativ: Exportieren, Datei oeffnen, pruefen ob Textgroessen und Abstaende passen, `width` / `height` oder `theme(base_size = ...)` anpassen, erneut exportieren. Diese Schleife ist effektiv, aber muehsam - und genau hier hilft camcorder.

> **Die Export-Pruef-Schleife beschleunigen**
>
> Zwei Tricks machen den iterativen Pruefzyklus schneller:
>
> 1. **Die Datei direkt aus R oeffnen** mit `shell.exec("my_plot.png")` (Windows) oder `browseURL("my_plot.png")` (plattformuebergreifend). Das spart den manuellen Schritt, die Datei im Dateimanager zu suchen.
> 2. **Waehrend der Entwicklung als PDF exportieren** und in einem Viewer oeffnen, der die Datei nicht sperrt. Sumatra PDF (Windows, kostenlos) laedt die Datei automatisch neu, sobald sie sich auf der Festplatte aendert - man kann `ggsave()` erneut ausfuehren und sieht den aktualisierten Plot sofort, ohne ihn schliessen und neu oeffnen zu muessen. Adobe Acrobat hingegen sperrt die Datei waehrend sie geoeffnet ist, sodass `ggsave()` mit einem “Datei wird verwendet”-Fehler fehlschlaegt.

Das camcorder-Paket

Das {camcorder}-Paket verfolgt einen anderen Ansatz: Anstatt zu exportieren und zu pruefen, fixiert es die RStudio-Viewer-Leinwand auf die exakten Exportabmessungen, *bevor* man mit dem Plotten beginnt. So stimmt die Vorschau mit dem Export ueberein - man sieht, was man bekommt.

```
camcorder::gg_record(
  device = "png",
  width = 6,
  height = 4,
  units = "in",
  dpi = 300
)
```

Nach einmaligem Ausfuehren von `gg_record()` zu Beginn einer Session wird jeder nachfolgende Plot im Viewer exakt mit 6 x 4 Zoll und 300 dpi gerendert. Die Abmessungen bleiben fest, unabhaengig von der Fenstergroesse. Man kann dann am Plot-Code iterieren - Schriftgroessen anpassen, Raender justieren, Legendenpositionen aendern - und das Ergebnis in Echtzeit in der finalen Exportgroesse sehen.

Wenn man fertig ist, speichert `gg_record()` auch eine Aufzeichnung jeder Plot-Version, die sich in ein animiertes GIF der Plot-Entwicklung verwandeln laesst:

```
camcorder::gg_playback(
  name = "plot_development.gif",
  first_image_duration = 3,
  last_image_duration = 8,
  frame_duration = 0.5
)
```

ggsave vs. camcorder

Beide Werkzeuge loesen das Canvas-Problem, aber auf unterschiedliche Weise:

- `ggsave()` ist eine einmalige Export-Funktion - Plot gestalten, dann exportieren. Am besten fuer schnelle Exports und geskriptete Pipelines.
- `camcorder` ist eine sitzungsweite Vorschau-Fixierung - Canvas zuerst fixieren, dann gestalten. Am besten fuer interaktive Entwicklung, bei der man iterativ an der Aesthetik arbeitet.

In der Praxis nutzen viele camcorder waehrend der Entwicklung und `ggsave()` in finalen Skripten.

Bibliography