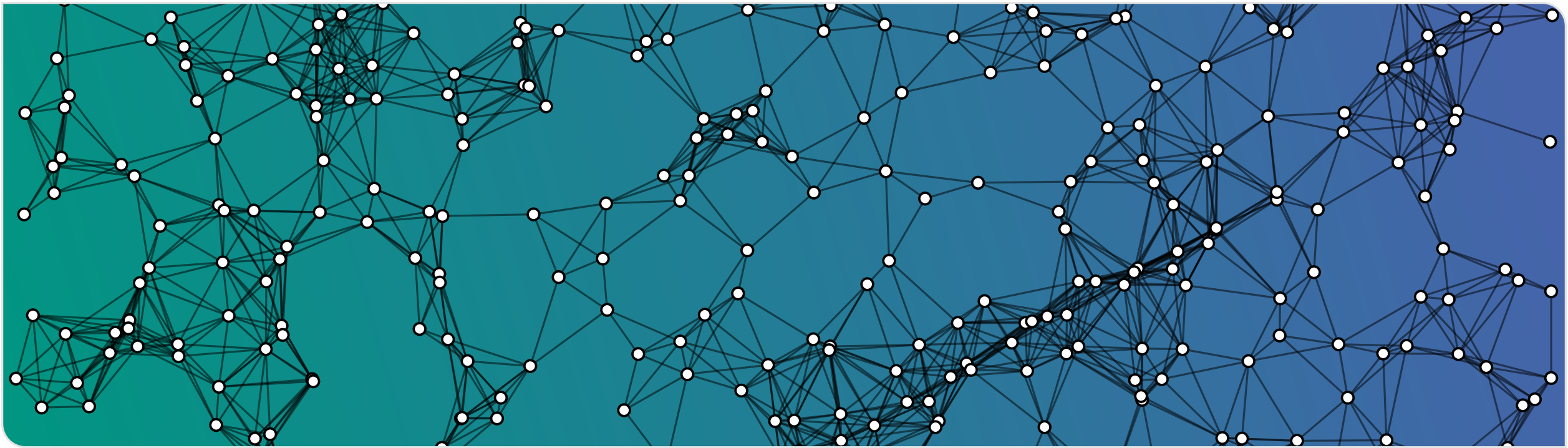
Karlsruher Institut für Technologie

Praktikum – Beating the Worst Case

Jean-Pierre von der Heydt und Marcus Wilhelm | 15.11.2023



Fragen zum Übungsblatt 0

Implementierung

- Wie seid ihr mit C++ klar gekommen?
- weitere Sprachen / Tools?
- habt ihr euer Auswertungs-setup automatisiert?

Aufgaben

- War klar, was die Aufgaben von euch verlangen?
- Wie schwer fandet ihr die Aufgaben?

Ergebnisse

- Konntet ihr die Netzwerke deutlich unterscheiden?
- Sind euch Graphparameter eingefallen?
- Habt ihr eine Vermutung, wie die Graphen erzeugt wurden?

Fragen zum Übungsblatt 0

Implementierung

- Wie seid ihr mit C++ klar gekommen?
- weitere Sprachen / Tools?
- habt ihr euer Auswertungs-setup automatisiert?

Aufgaben

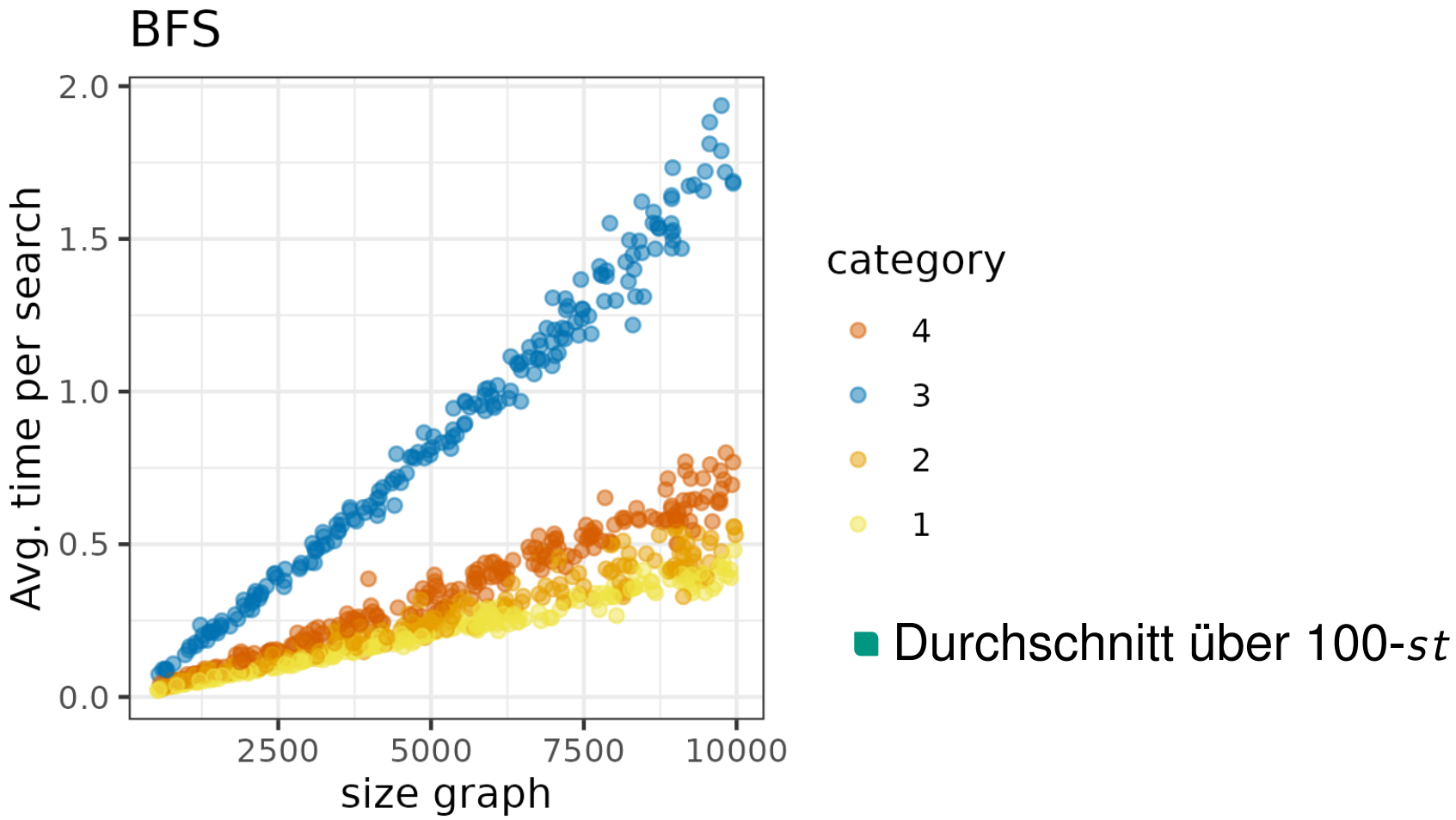
- War klar, was die Aufgaben von euch verlangen?
- Wie schwer fandet ihr die Aufgaben?

Auswertung:
Jonas, Philipp, Kilian: 400P
Sven, David: 400P (:
Victoria, Cedrico: 400P

Ergebnisse

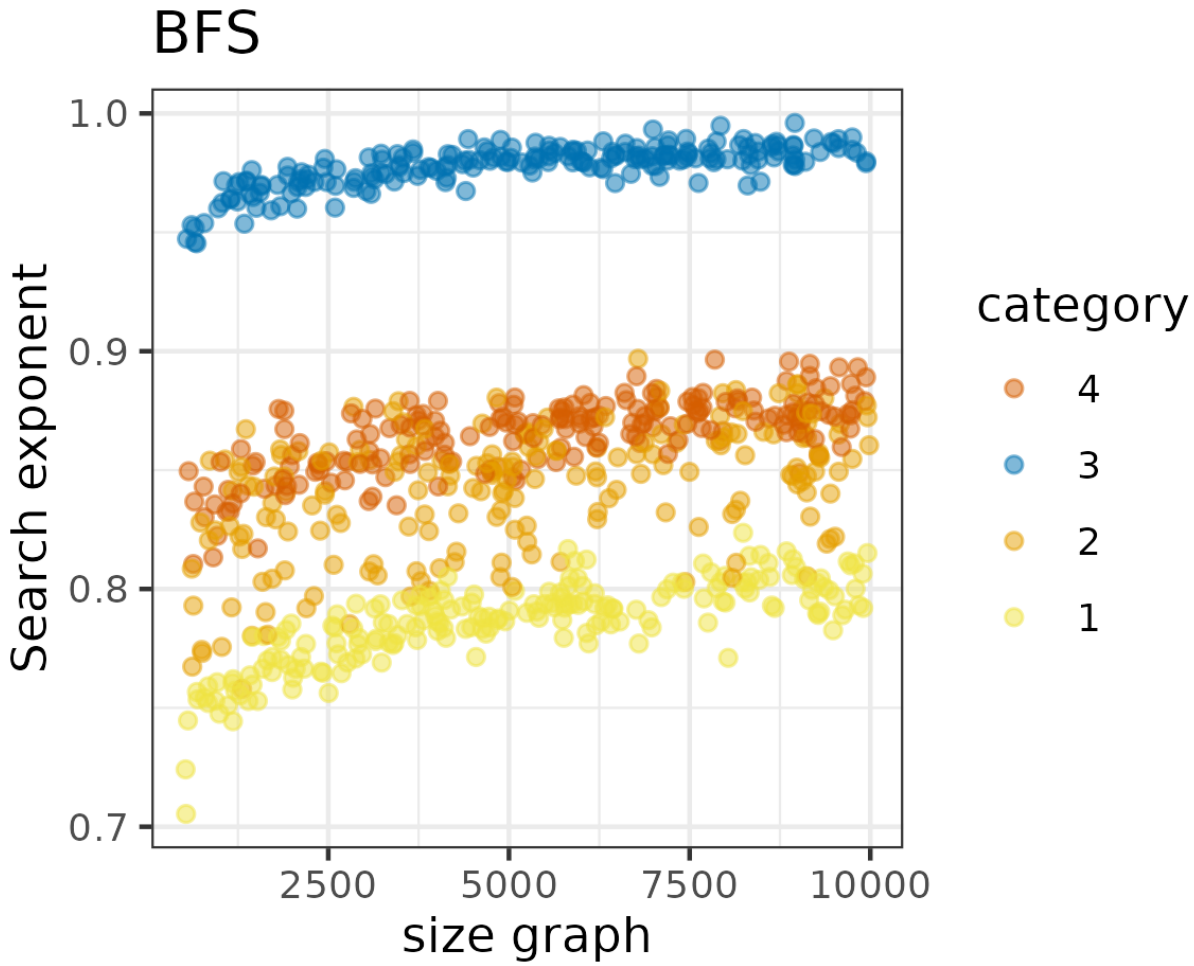
- Konntet ihr die Netzwerke deutlich unterscheiden?
- Sind euch Graphparameter eingefallen?
- Habt ihr eine Vermutung, wie die Graphen erzeugt wurden?

Lösung Übungsblatt 0



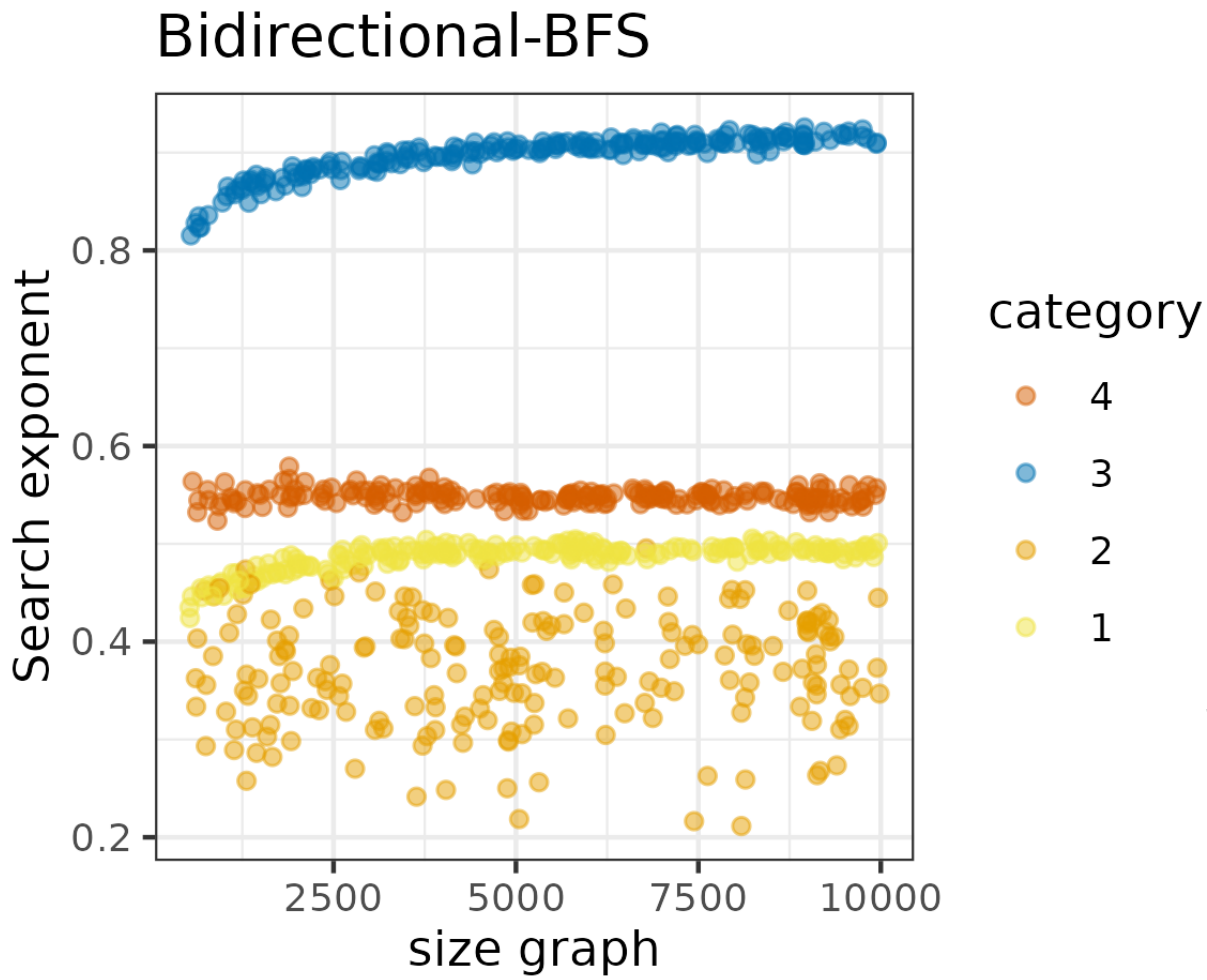
- Durchschnitt über 100-st

Lösung Übungsblatt 0



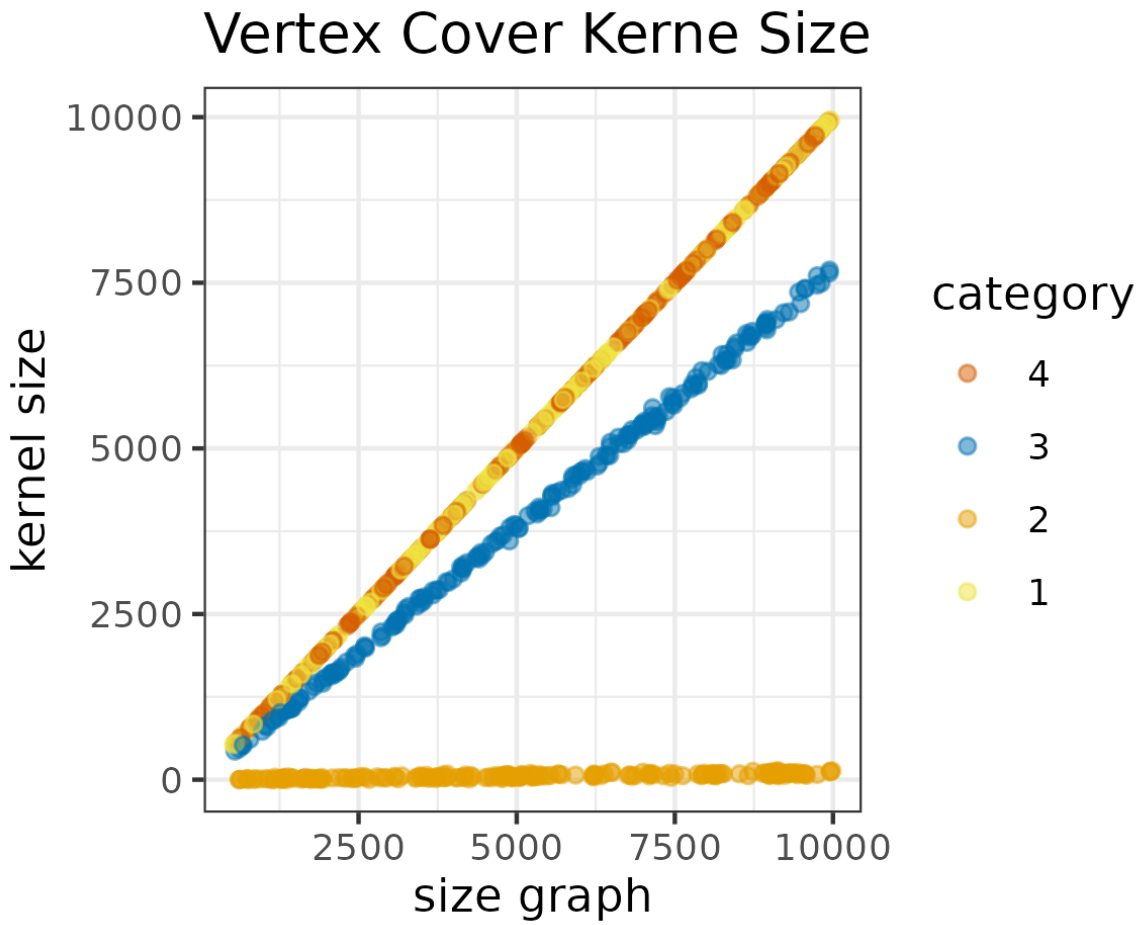
Für druchschnittliche Suchraumgröße s, berechne x mit $s = m^x$
$\Rightarrow x = \log_m(s)$

Lösung Übungsblatt 0

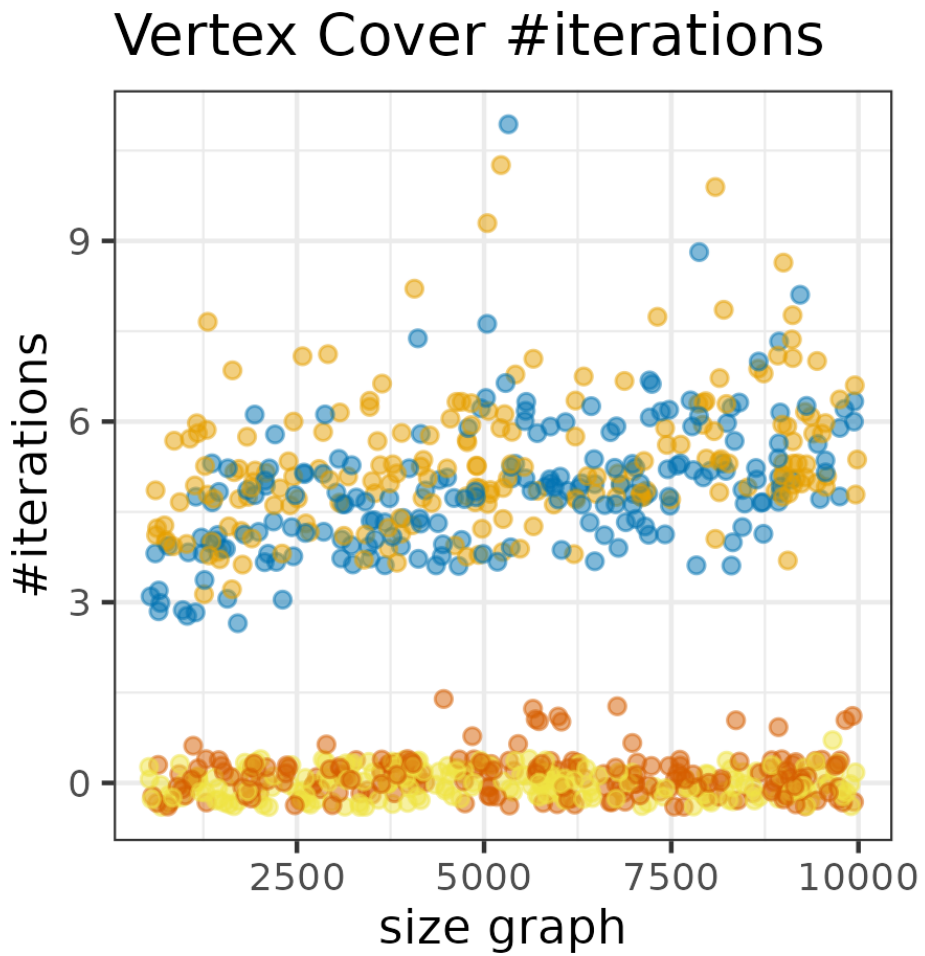


Für druchschnittliche Suchraumgröße s, berechne x mit $s = m^x$
$\Rightarrow x = \log_m(s)$

Lösung Übungsblatt 0



Lösung Übungsblatt 0

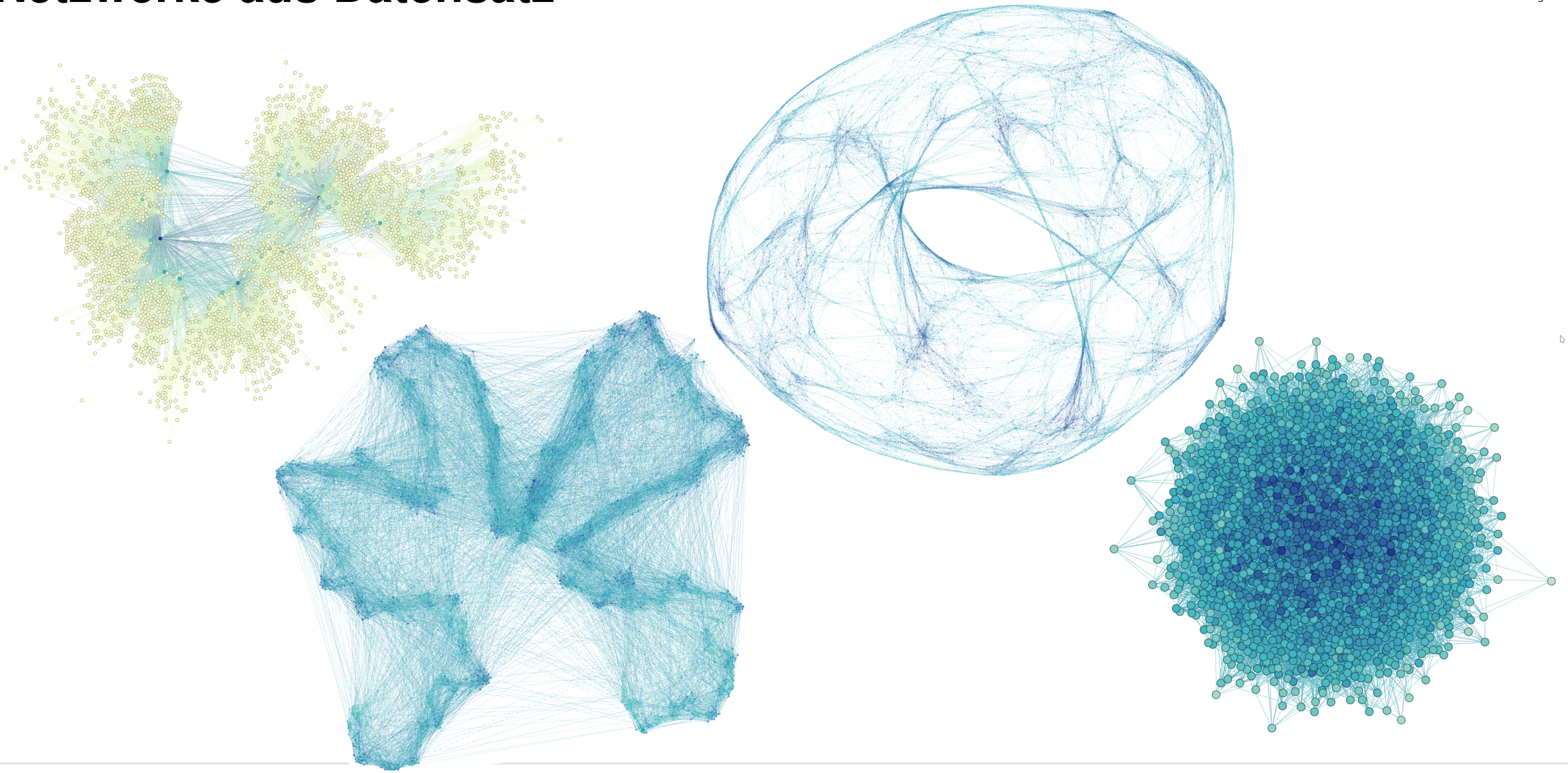


- jeweils ∼8min für das BFS/VC Experiment

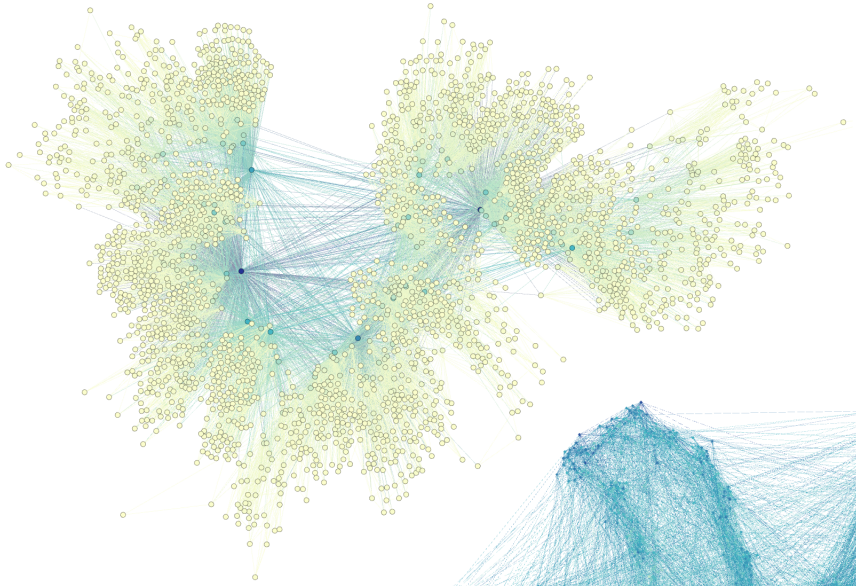
Wichtige Lösungsdetails

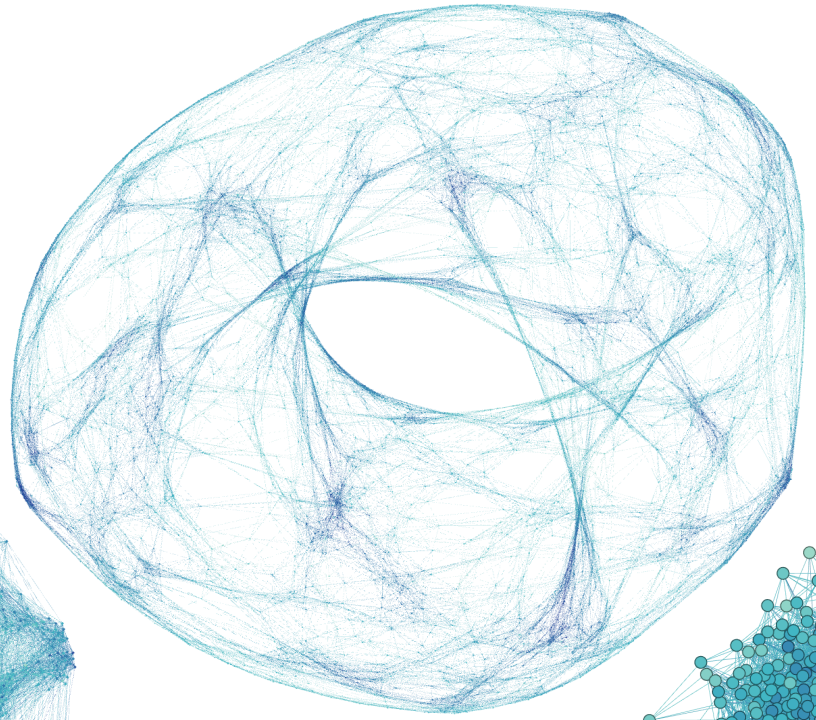
- Zeitmessungen können viel Rauschen erzeugen
  - Messungs wiederholen und Mitteln
  - Suchraumgröße statt Zeit messen
- Messwerte Visualisieren
  - hilft Fehler oder Muster zu erkennen
- auf effiziente Implementierung achten
  - $\sim$1Mio. Operationen pro Sekunden
  - bei 10000 Knoten kein $\mathcal{O}(n^2)$
- hilfreich: automatisiertes Experiment Set-up

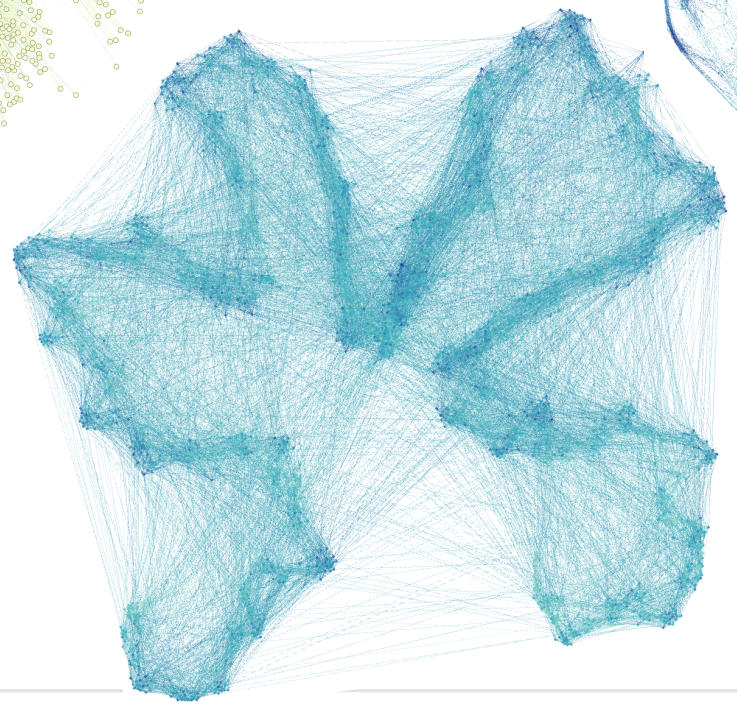
Netzwerke aus Datensatz

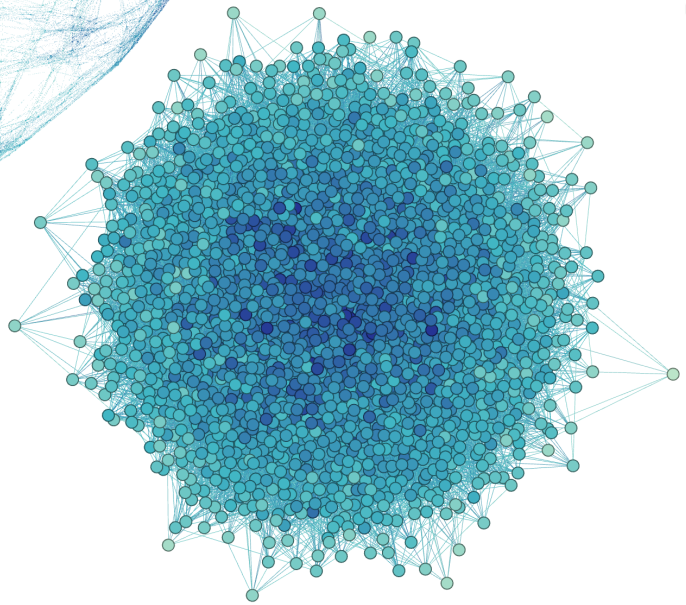


Netzwerke aus Datensatz



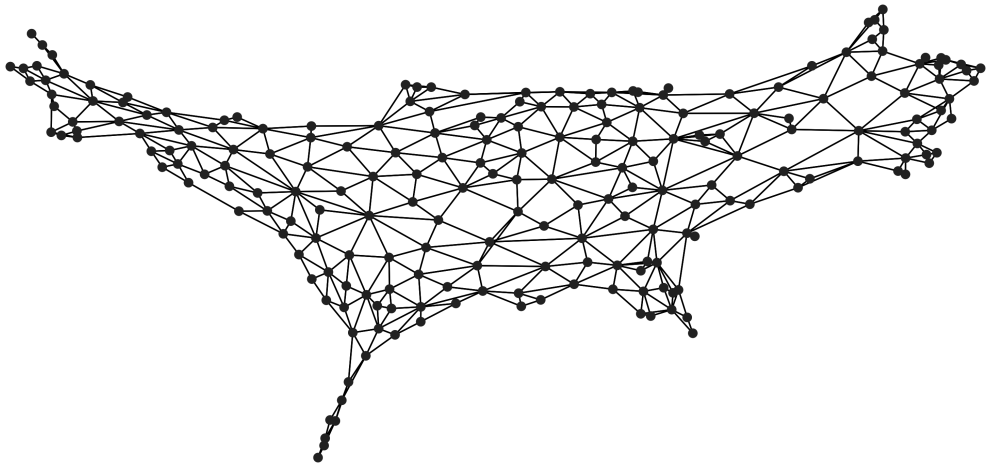




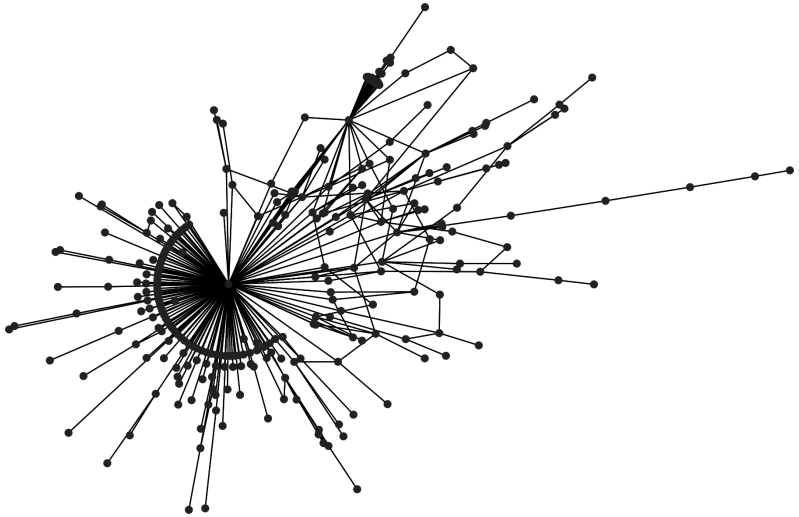


Motivation Blatt 1:
Unterschiede verstehen & quantifizieren

Gradverteilung

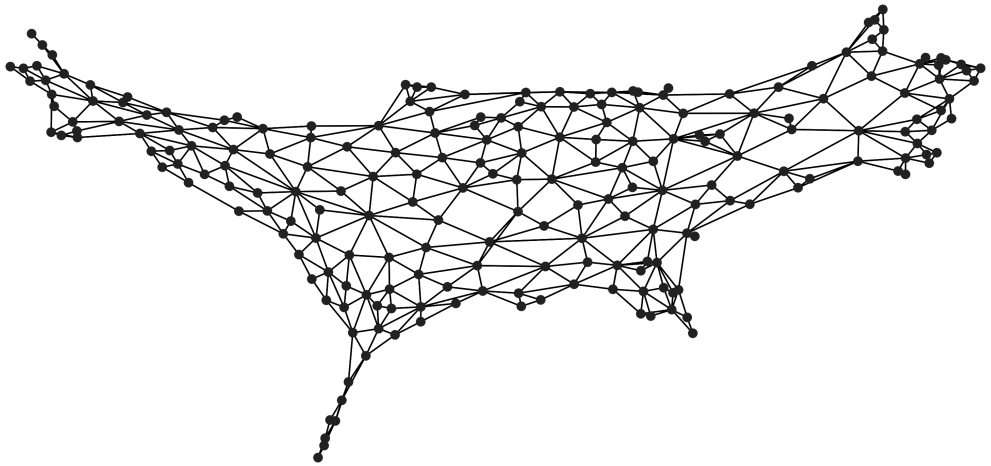


- Teil des US-Straßennetzes

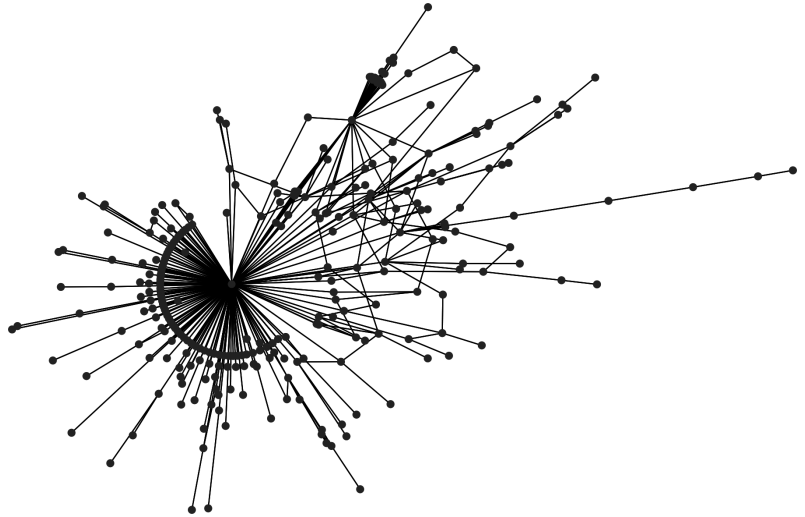


- Ausschnitt von Twitter Retweets

Gradverteilung

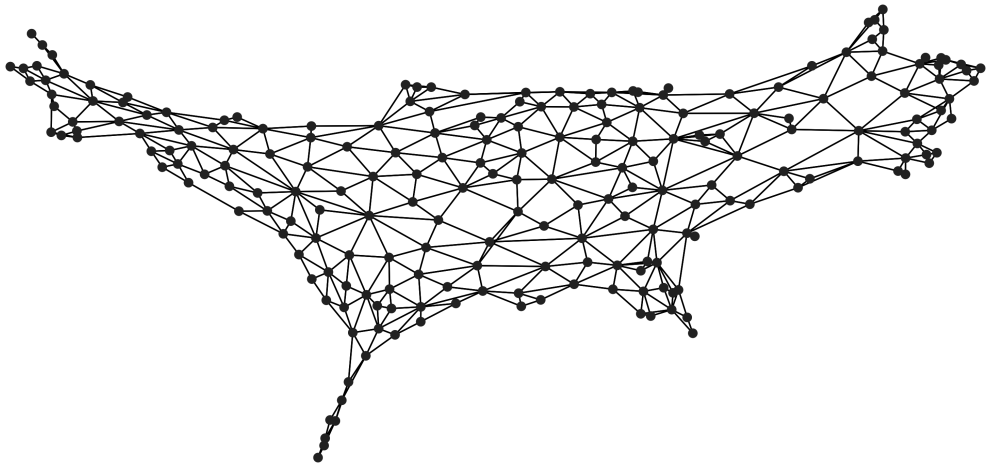


- Teil des US-Straßennetzes
- alle Knoten haben ähnlichen Grad

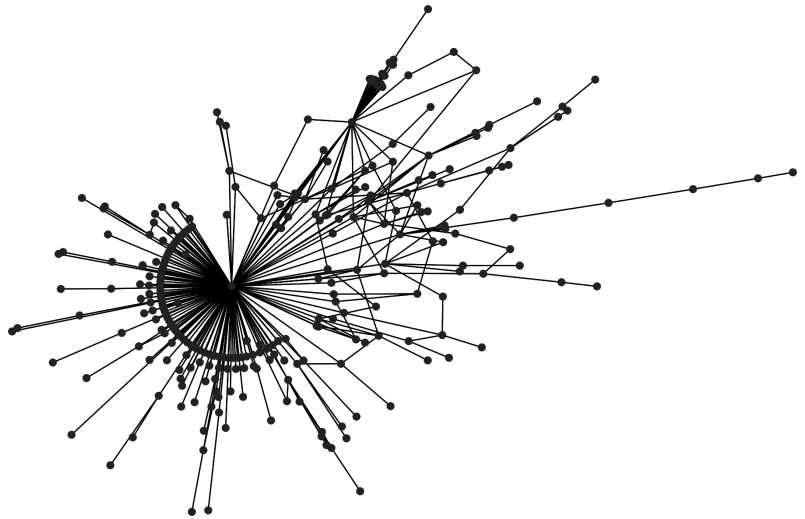


- Ausschnitt von Twitter Retweets
- zwei/drei extrem hochgradige Knoten

Gradverteilung



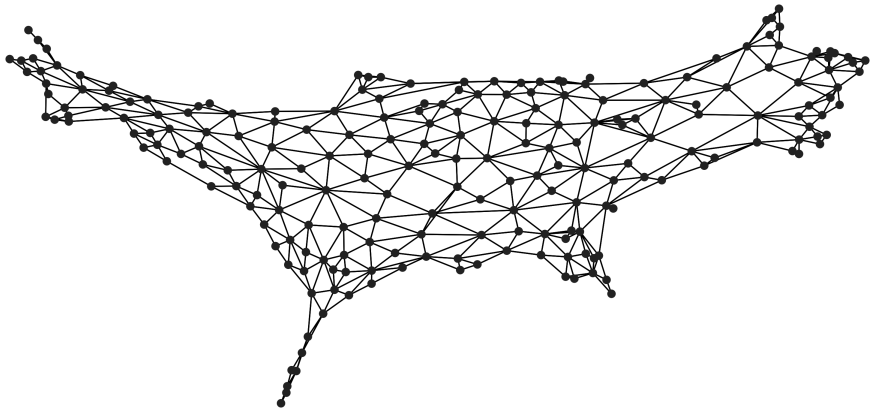
- Teil des US-Straßennetzes
- alle Knoten haben ähnlichen Grad



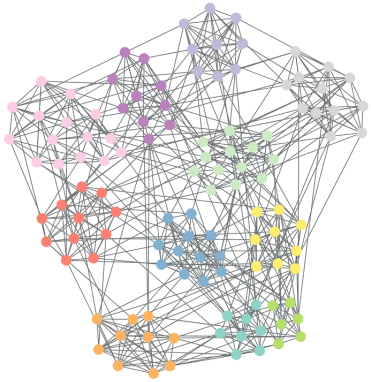
- Ausschnitt von Twitter Retweets
- zwei/drei extrem hochgradige Knoten

- Durchschnitts-/Maximalgrad nicht aussagekräftig genug
- konkrete Verteilung der Knoten ist relevant

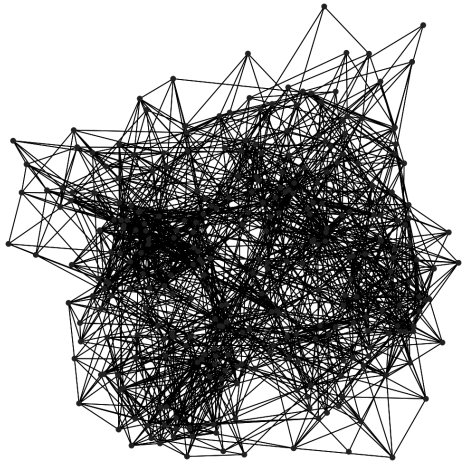
Lokalität



- Teil des US-Straßennetzes
- Knoten mit gemeinsamen Nachbarn sind oft selbst benachbart

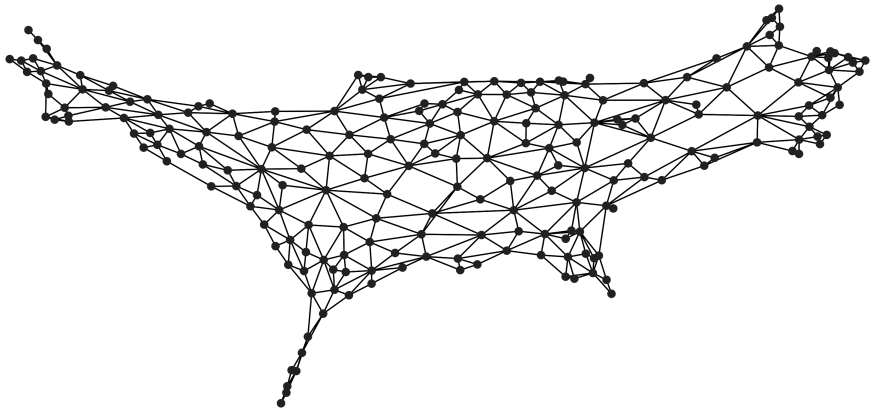


- Spiele von College football teams
- Kanten entsprechen community structure

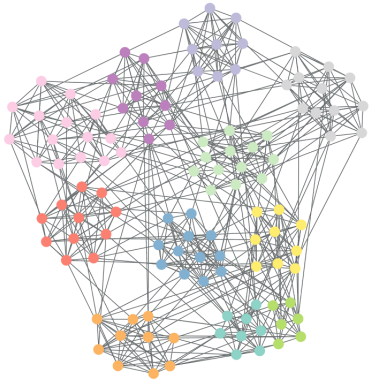


- Erdős–Rényi Graph
- Jede mögliche Kante existiert mit Wahrscheinlichkeit p (unabhängig)

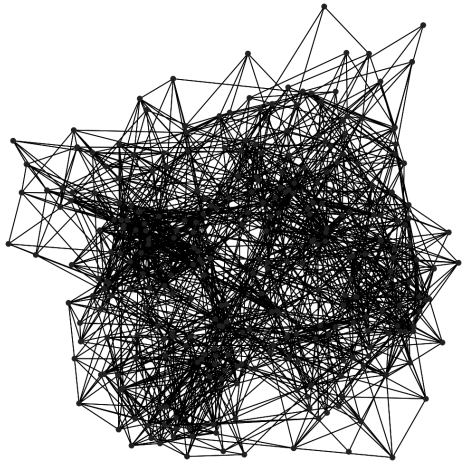
Lokalität



- Teil des US-Straßennetzes
- Knoten mit gemeinsamen Nachbarn sind oft selbst benachbart

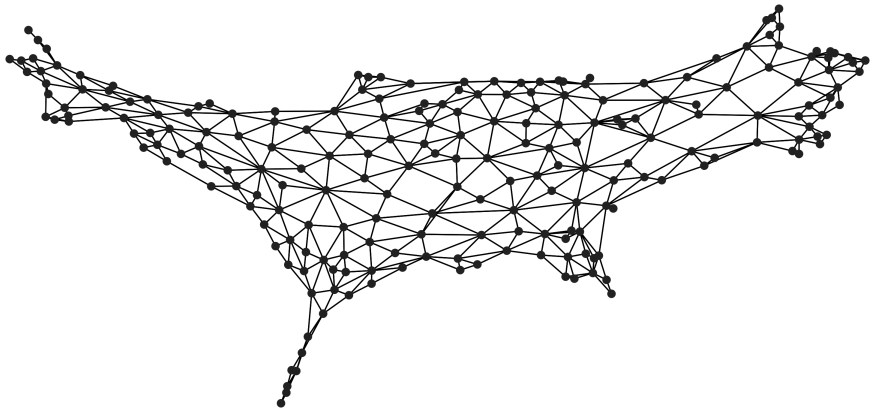


- Spiele von College football teams
- Kanten entsprechen community structure

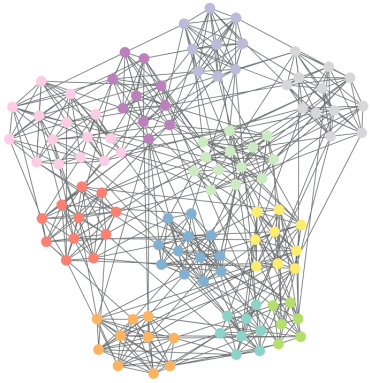


- Erdős–Rényi Graph
- Jede mögliche Kante existiert mit Wahrscheinlichkeit p (unabhängig)

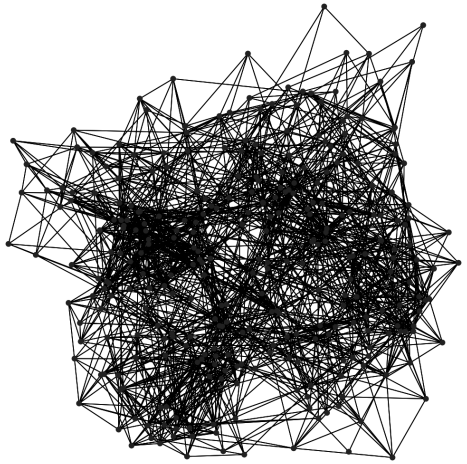
Lokalität



- Teil des US-Straßennetzes
- Knoten mit gemeinsamen Nachbarn sind oft selbst benachbart



- Spiele von College football teams
- Kanten entsprechen community structure



- Erdős–Rényi Graph
- Jede mögliche Kante existiert mit Wahrscheinlichkeit p (unabhängig)

- *„Lokalität“*, *„Clustering“*, *„Geometrie“*
- In vielen Echtwelt Netzwerken beobachtet (z.B. Straßen-, soziale Netzwerke)

- keine „Lokalität“
- high *“temperature”*

Workflow

Motivation: Erweiterbarkeit, Wiederholbarkeit

Workflow

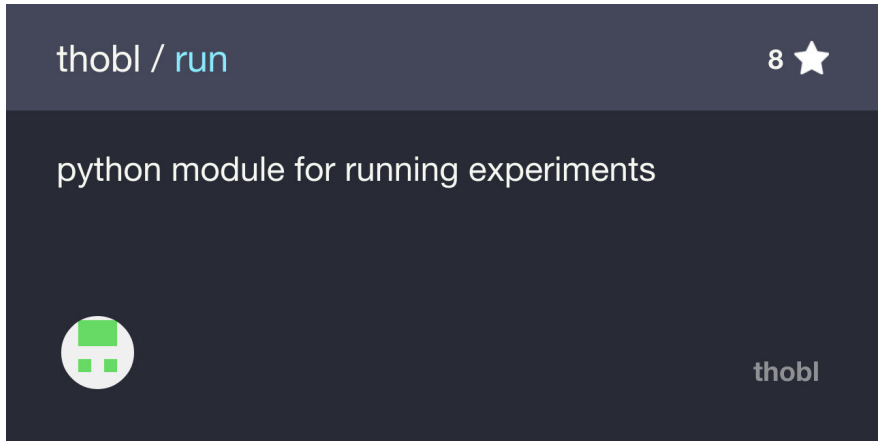
Motivation: Erweiterbarkeit, Wiederholbarkeit

Beispiel:
- neue Algorithmen
- neue Eingaben
- neue Messwerte

Workflow

Motivation: Erweiterbarkeit, Wiederholbarkeit

Beispiel:
- neue Algorithmen
- neue Eingaben
- neue Messwerte

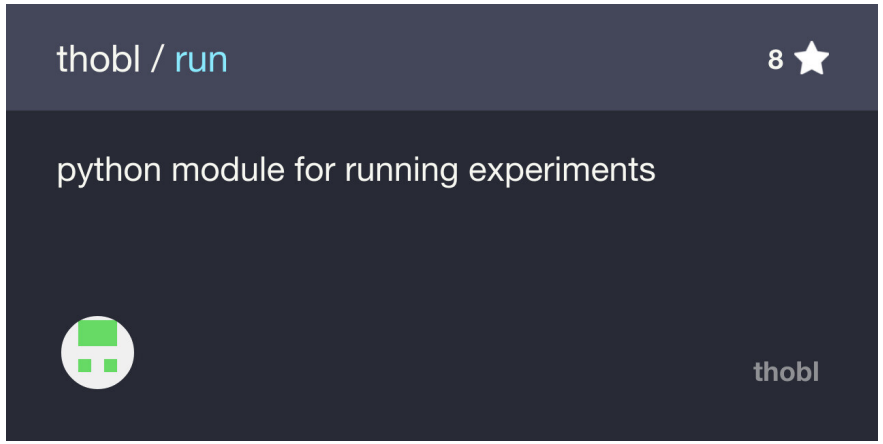


`https://github.com/thobl/run`

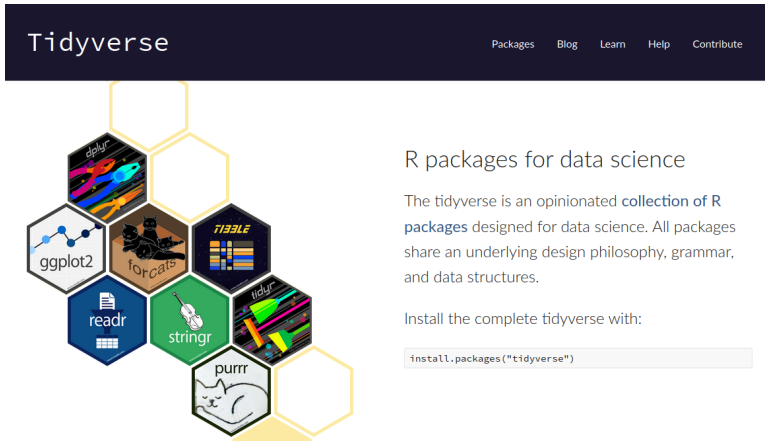
Workflow

Motivation: Erweiterbarkeit, Wiederholbarkeit

Beispiel:
- neue Algorithmen
- neue Eingaben
- neue Messwerte



`https://github.com/thobl/run`



Plotting with R

Run

```
# example.py
import run

run.add(
    "experiment1",
    "mycommand [[file]] -x [[param1]] -y [[param2]]",
    {'file': ["file1", "file2", "file3"],
     'param1': [1, 2, 3, 4],
     'param2': [8, 16, 32]},
    stdout_file="output/[[file]]_x=[[param1]]_y=[[param2]].txt"
)

run.use_cores(8)
run.run()
```

Run

```
import run
from glob import glob
from os.path import basename

input_files = glob("input/*.gr")

run.group("Stats")
run.add(
    "stats",
    "build/cli/stats --noheader [[input_files]]",
    {'input_files': input_files},
    stdout_file="output/stats.csv",
    header_command="build/cli/stats /dev/null --onlyheader",
)

run.group("BiBFS")
run.add(
    "bibfs_cost",
    "build/cli/bfs -s 100 --noheader [[input_files]]",
    {'input_files': input_files},
    stdout_file=lambda args: f"output/bfs/{basename(args['
      input_files'])}.csv",
)


run.group("Postprocessing")
# add run to merge csvs

run.use_cores(4)
run.run()
```

Run

```
import run
from glob import glob
from os.path i

input_files =

run.group("Sta
run.add(
    "stats",
    "build/cli
    {'input_fi
    stdout_fil
    header_com
)

run.group("BiB
run.add(
    "bibfs_cost",
    "build/cli/bfs -s 100 --noheader [[input_files]]",
    {'input_files': input_files},
    stdout_file=lambda args: f"output/bfs/{basename(args['
      input_files'])}.csv",
)


run.group("Postprocessing")
# add run to merge csvs

run.use_cores(4)
run.run()
```

```
marcus @ i11pcwilhelm in ~/work/teaching/praktikum_beating_the_worst_case_framework [18:41:44]
>>> python experiment.py BiBFS                                                    ±[●][master]
                    todo   skipped     total
Stats
 └─ stats              0       800       800
BiBFS
 └─ bibfs_cost       630       170       800
Postprocessing
 └─ merge_csvs         0         1         1

running the experiments:
bibfs_cost:   5%|▌                                          | 34/630 [00:08<03:35,  2.77it/s]
```

Plotting with R

Was ist R?

- freie, open-source Programmiersprache
- primär für Statistik und Grafiken
- viele nützliche Pakete, z.B. ggplot2 und tidyr

ggplot2

- sehr umfangreiches und flexibels Paket für plotting
- basiert of "Grammar of Graphics"
  - sehr Strukturiert, hohe Abstraktion
  - komplexe Grafiken lassen sich leicht erstellen

Tidyverse, tidyr, dplyr

- Paket für Datentransformation
- strukturiert, elegant und einfach

Plotting with R – Example

```
library(dplyr)
library(tidyr)
library(ggplot2)

stats <- tibble(read.csv("../output/stats.csv"))
bfs <- tibble(read.csv("../output/bfs_merged.csv"))

# join using column 'graph'
data <- bfs %>%
    inner_join(stats, by = "graph")
```

Plotting with R – Example

```
library(dplyr)
library(tidyr)
library(ggplot2)

stats <- tibble(read.csv("../output/stats.csv"))
bfs <- tibble(read.csv("../output/bfs_merged.csv"))

# join using column 'graph'
data <- bfs %>%
    inner_join(stats, by = "graph")
```

Zuweisungsoperator

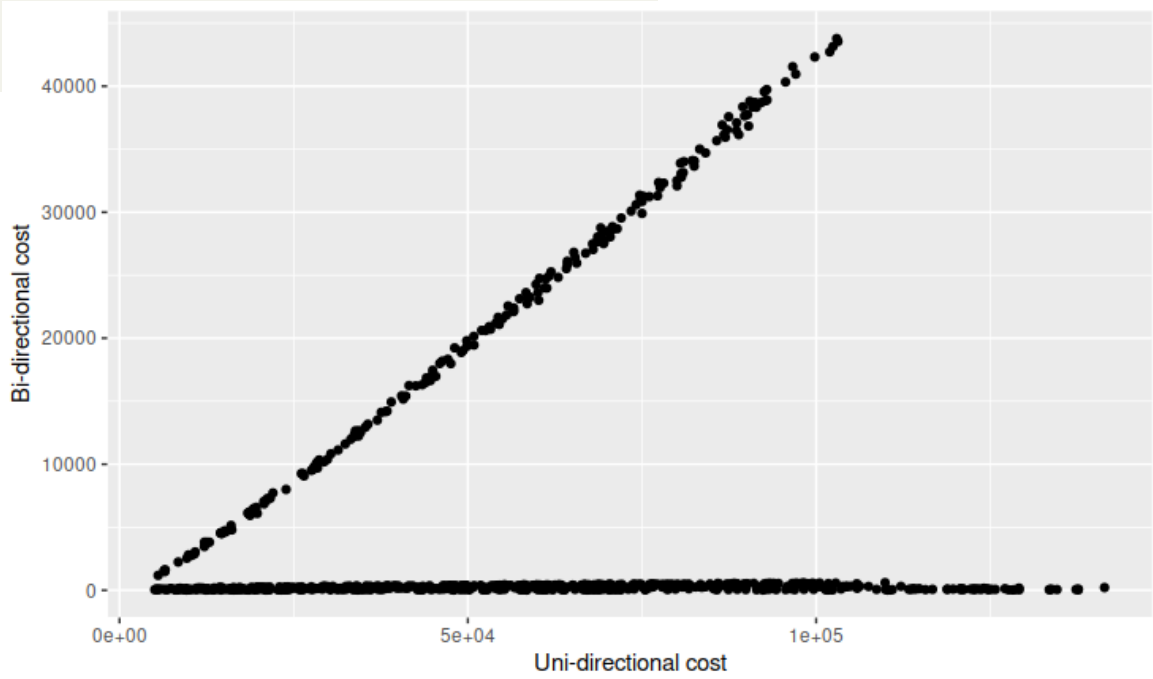
Pipe %>% äquivalent zu:
`inner_join(bfs, stats, ...)`

Plotting with R – Example

```
library(dplyr)
library(tidyr)
library(ggplot2)

stats <- tibble(read.csv("../output/stats.csv"))
bfs <- tibble(read.csv("../output/bfs_merged.csv"))

# compute mean per graph
bfs <- bfs %>%
    group_by(graph) %>%
    summarise(
        dist = mean(dist),
        cost_uni = mean(cost_uni),
        cost_bi = mean(cost_bi)
    )

# join using column 'graph'
data <- bfs %>%
    inner_join(stats, by = "graph")
```

Plotting with R – Example

```
# table bfs as before

bfs %>%
  ggplot(aes(x = cost_uni, y = cost_bi)) +
  geom_point() +
  xlab("Uni-directional cost") +
  ylab("Bi-directional cost")
```



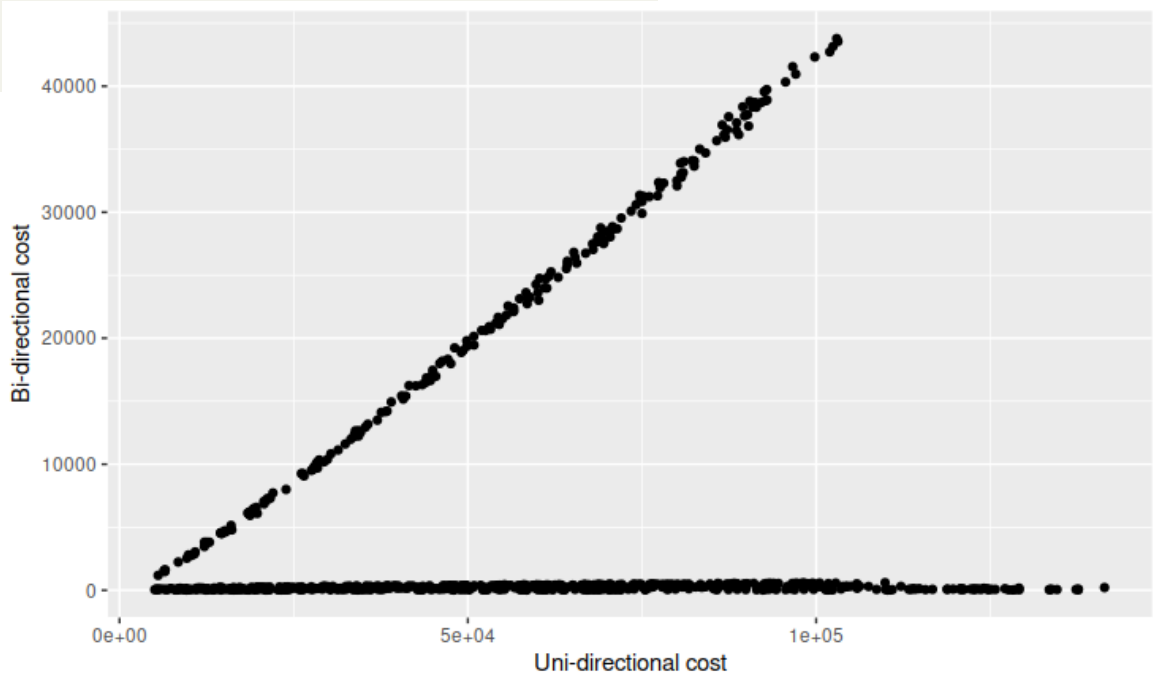
Plotting with R – Example

```
# table bfs as before

bfs %>%
  ggplot(aes(x = cost_uni, y = cost_bi)) +
  geom_point() +
  xlab("Uni-directional cost") +
  ylab("Bi-directional cost")
```

weitere Attribute: color, shape, size, etc.

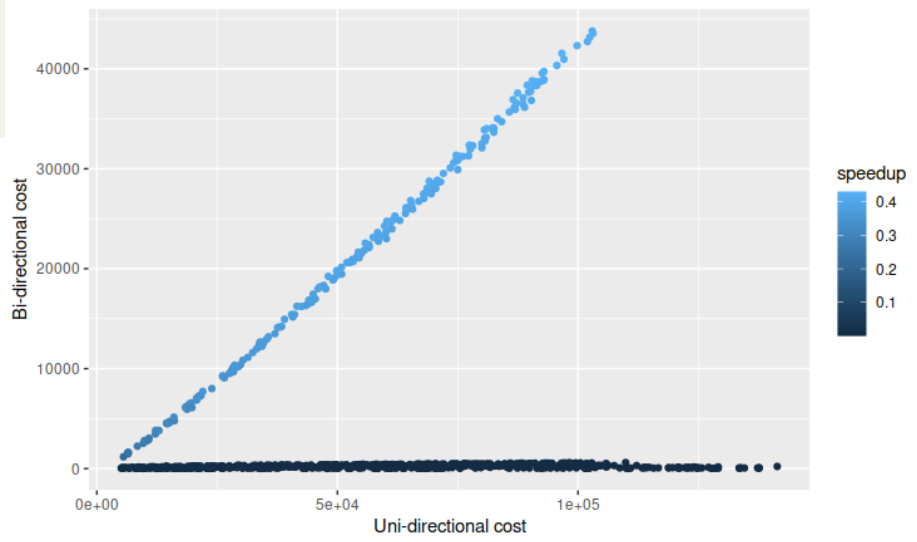
geom_line, geom_barplot, ...

scale_y_log10, facet_grid ...



Plotting with R – Example

```
# table bfs as before

bfs <- bfs %>%
  mutate(
    speedup = cost_bi / cost_uni
  )

bfs %>%
  ggplot(aes(x = cost_uni, y = cost_bi, color=speedup)) +
  geom_point() +
  xlab("Uni-directional cost") +
  ylab("Bi-directional cost")
```

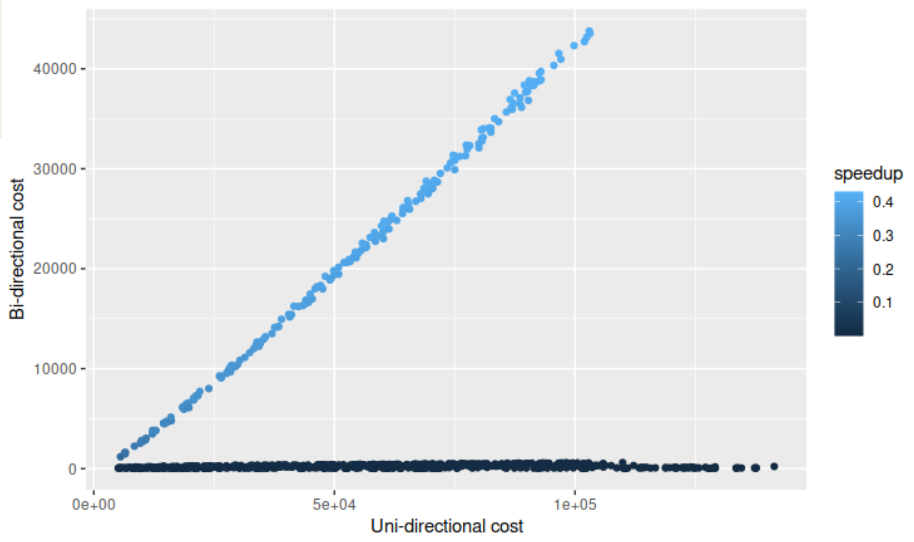


Plotting with R – Example

```
# table bfs as before

bfs <- bfs %>%
  mutate(
    speedup = cost_bi / cost_uni
  )

bfs %>%
  ggplot(aes(x = cost_uni, y = cost_bi, color=speedup)) +
  geom_point() +
  xlab("Uni-directional cost") +
  ylab("Bi-directional cost")
```

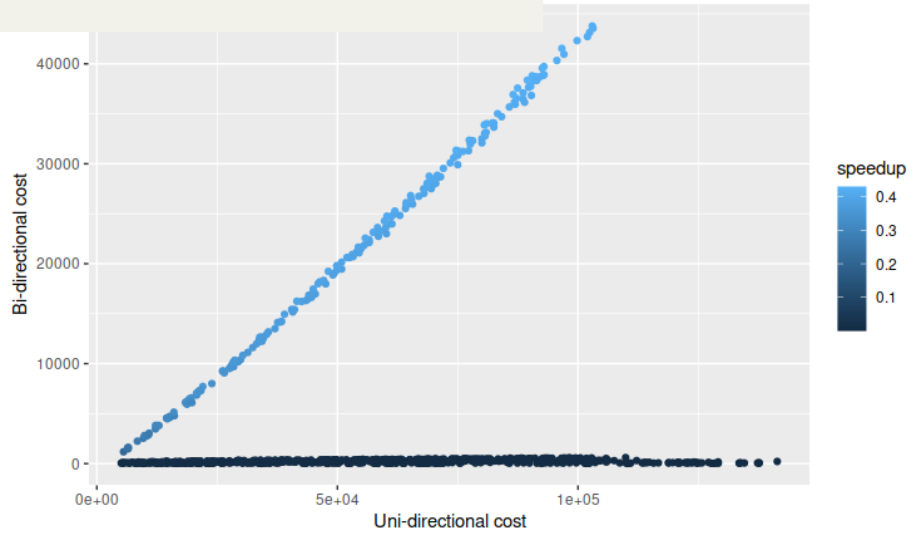
Neben `mutate` auch hilfreich:

- `group_by`, `summarize`
- `pivot_longer`, `pivot_wider`



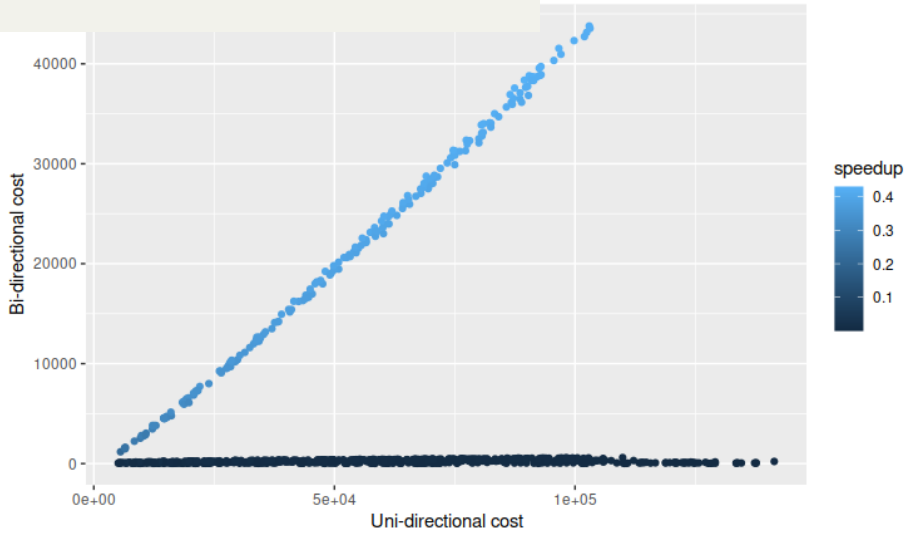
Plotting with R – Example

```
# table bfs as before

bfs %>%
  mutate(
    speedup = cost_bi / cost_uni
  ) %>%
  ggplot(aes(x = cost_uni, y = cost_bi, color=speedup)) +
  geom_point() +
  xlab("Uni-directional cost") +
  ylab("Bi-directional cost")
```



Plotting with R – Example

```
# table bfs as before

bfs %>%
  mutate(
    speedup = cost_bi / cost_uni
  ) %>%
  ggplot(aes(x = cost_uni, y = cost_bi, color=speedup)) +
  geom_point() +
  xlab("Uni-directional cost") +
  ylab("Bi-directional cost")
```



Nützliche Resourcen:

- R for Data Science, Chapter 1: Data visualization
- Tutorial Website
- Beispielcode anpassen
- Editor: RStudio, emacs
- `ggplot2` Cheat Sheet