Datenmanipulation mit „plyr, dplyr, reshape2“

Hung Quan Vu

Fachbereich Informatik

Betreuer: Jakob Lüttgau

2016 - 05 -04

- PLYR
- DPLYR
- Reshape2

- Was ist PLYR?
  - Ist ein R Paket das vereinfacht den
    - Dateisaufteilungsvorgang
    - Dateisbearbeitungsvorgang
    - Dateisvereinigungsvorgang.

  - Ist sehr effektiv wenn wir brauchen zb
    - passen das gleiche Modell für jeder subsets eines Dataframe
    - zusammenpassende Statistiken für jede Gruppe schnell berechnen
    - Führen gruppenweise Transformationen wie Skalieren oder Standardisierung

- Warum wollen wir PLYR benutzen ?
  - völlig konsistente Namen, Argumente und Ausgänge
  - bequem Parallelisierung durch die foreach-Paket
  - Eingabe von und Ausgabe an data.frames, Matrizen und Listen
  - Fortschrittsbalken für lang laufenden Betrieb zu verfolgen
  - eingebaute Fehlerwiederherstellung und informative Fehlermeldungen
  - Etiketten, die für alle Transformationen beibehalten werden
  - PLYR > Base R Apply Function > Schleife (for)

- Die Basis
  - Die Basisformatte ist 2 Buchstaben und dann ply().
    - Die erste Buchstabe steht für Eingabe
    - Die zweite Buchstabe steht für Ausgabe
  - Die 3 Hauptbuchstaben sind
    - d = data frame
    - a = array (enthält Matrizen)
    - l = list
    - zb ddply, ldply, alply, aaply,...
  - Einige weniger häufig Buchstaben: m, r, _

- Ein allgemein Beispiel mit PLYR

  - Wir nehmen ein einfach Data frame

    - durch Jahre aufteilen
    - Koeffizient von Variation von der Zählung berechnen
    - Ein Data frame zurückliefern

Abbildung 01
Quelle:“textbroker“ [Ab01]

- Ein allgemein Beispiel mit PLYR

```
> set.seed(1)
> d <- data.frame(year = rep(2000:2002, each = 3),+ count = round(runif(9, 0, 20)))
> print(d)
```

	Year	Count
1	2000	5
2	2000	7
3	2000	11
4	2001	18
5	2001	4
6	2001	18
7	2002	19
8	2002	13
9	2002	13

Beispiel 01
Quelle:“cran.r-project.org“
[Bsp01]

- Ein allgemein Beispiel mit PLYR

```
> library(plyr)

> ddply(d, "year", function(x) {+ mean.count <- mean(x$count)+ sd.count <- sd(x$count)+ cv <-
sd.count/mean.count+ data.frame(cv.count = cv)+ })
```

	Year	Count
1	2000	0.3984848
2	2001	0.6062178
3	2002	0.2309401

Beispiel 01
Quelle:“cran.r-project.org“
[Bsp01]

- Transform und Summarise
  - Transformation
    - wirkt als wäre es normalerweise als Funktionsbasis R
    - ändert einen bestehenden Dataframe .
  - Summarise erstellt eine neue (in der Regel) verkürzte Dataframe .

- Transform und Summarise

```
> ddply(d, "year", summarise, mean.count = mean(count))
```

	Year	Mean.count
1	2000	7.666667
2	2001	13.333333
3	2002	15.000000

Beispiel 02
Quelle:“cran.r-project.org“
[Bsp02]

- Transform und Summarise

```
> ddply(d, "year", transform, total.count = sum(count))
```

	Year	Count	Total count
1	2000	5	23
2	2000	7	23
3	2000	11	23
4	2001	18	40
5	2001	4	40
6	2001	18	40
7	2002	19	45
8	2002	13	45
9	2002	13	45

Beispiel 02

Quelle:“cran.r-project.org“

[Bsp02]

- Transform und Summarise
  - Bonus function: mutate.

```
> ddply(d, "year", mutate, mu = mean(count), sigma = sd(count),+ cv = sigma/mu)
```

	Year	Count	mu	sigma	cv
1	2000	5	7.666667	3.055050	0.3984848
2	2000	7	7.666667	3.055050	0.3984848
3	2000	11	7.666667	3.055050	0.3984848
4	2001	18	13.333333	8.082904	0.6062178
5	2001	4	13.333333	8.082904	0.6062178
6	2001	18	13.333333	8.082904	0.6062178
7	2002	19	15.000000	3.464102	0.2309401
8	2002	13	15.000000	3.464102	0.2309401
9	2002	13	15.000000	3.464102	0.2309401

Beispiel 02

Quelle:“cran.r-project.org“ [Bsp02]

- Nested chunking of the data
  - Die grundlegende Syntax kann leicht erweiteret werden, um die Datei auseinander auf mehrere Spalten zu brechen

```
> baseball.dat <- subset(baseball, year > 2000) # data aus dem plyr Paket
> x <- ddply(baseball.dat, c("year", "team"), summarize,+ homeruns = sum(hr))
> head(x)
```

	year	Team	homeruns
1	2001	ANA	4
2	2001	ARI	155
3	2001	ATL	63
4	2001	BAL	58
5	2001	BOS	77
6	2001	CHA	63

Beispiel 03
Quelle:“inside-r.org“
[Bsp03]

- Andere nutzlich Funktion :

  - Fehler handeln

```
> f <- function(x) if (x == 1) stop("Error!") else 1
> safe.f <- failwith(NA, f, quiet = TRUE)
> #llply(1:2, f)
> llply(1:2, safe.f)

[[1]]
[1] NA

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

- Andere nutzlich Funktion :
  - Parallel processing

> x <- c(1:10) > wait <- function(i) Sys.sleep(0.1) > system.time(llply(x, wait))	> system.time(sapply(x, wait))	> library(doMC) > registerDoMC(2) > system.time(llply(x, wait, .parallel = TRUE))
user system elapsed 0.001 0.000 1.007	user system elapsed 0.001 0.000 1.010	user system elapsed 0.021 0.006 0.533

Beispiel 04
Quelle:“inside-r.org“
[Bsp04]

- Ist PLYR unbesiegbar ?
  - PLYR ist nicht immer schnell
- Lösungen ?

Abbildung 02
Quelle:“textbroker“ [Ab02]

- Lösung 1
  - Basis R-apply-Funktion verwenden

```
> system.time(ddply(baseball, "id", summarize, length(year)))

User    system elapsed
1.169   0.013   1.188

> system.time(tapply(baseball$year, baseball$id, function(x) length(x)))

User    system elapsed
0.019   0.000   0.020
```

Abbildung 03
Quelle:“blaublog“ [Ab03]

Beispiel 05
Quelle:“r-bloggers.com“
[Bsp05]

- Lösung 2
  - Immutable data frame verwenden

```
> system.time(ddply(idata.frame(baseball), "id", summarize, length(year)))

User   system elapsed
0.956   0.003    0.960
```

Beispiel 06
Quelle:“r-bloggers.com“
[Bsp06]

Abbildung 03
Quelle:“blaublog“ [Ab03]

- Lösung 3
  - Data.table Paket verwenden

```
> library(data.table)
> dt <- data.table(baseball, key="id")
> system.time(dt[, length(year), by=list(id)])

user   system elapsed
0.005  0.000   0.005
```

Beispiel 07
Quelle:“r-bloggers.com“
[Bsp07]

Abbildung 03
Quelle:“blaublog“ [Ab03]

Fragen?

Abbildung 04
Quelle:“textbroker“ [Ab04]

- Was ist DPLYR
  - DPLYR ist die nächste Iteration von PLYR
  - Ziel ist nur Data frames
  - DPLYR ist schneller als PLYR, hat mehr konsistent API

- Warum wollen wir DPLYR benutzen ?
  - Zeit ist Geld (dplyr ist bis zu 1000 mals schneller als plyr)
  - dplyr erleichtert dies durch einzelne Funktionen aufweisen, die zu den häufigsten Operationen entsprechen.
  - Jede Funktion löst nur eine einzige Sache.

- Verbesserung
  - einzelnen Funktionen entsprechen häufigsten Operationen
  - Jede Funktion ist optimiert um nur einen Problem effizient zu lösen
  - Verkettungs Befehle mit %.% (%>%)
  - viel schneller Berechnungen
  - alles was man zu einem lokalen Dataframe tun kann kann man auch zu einem Remote-Datenbanktabelle tun(zum Beispiel MySQL, SQLite)

- Manipuliertsfunktionen
  - filter: subset Zeilen. Mehrere Bedingungen kombiniert von & nur; | ist nicht verfügbar.
  - select : subset Spalten. Mehrere Spalten könne zurückgegeben werden.
  - arrange: Zeilen umordnen. Bietet Platz für mehrere Eingänge und auf- / absteigend Ordnung.
  - mutate: fügt neue Spalten hinzu, möglicherweise auf der Grundlage anderer Spalten; mehrere Eingänge erstellen mehrere Spalten.
  - summarize: jede Funktion innerhalb der Gruppen zu berechnen, so dass jede Gruppe auf eine einzige Zeile reduziert. Mehrere Eingänge erstellen mehrere Ausgabe summarize.

- Chaining commands

  - Nested R-Befehle sind oft schwierig zu lesen, weil
    - die Reihenfolge der Operationen aus dem innersten zu den äußersten Funktionen zu lesen.
    - Folglich treten die Argumente für diese äußerste Funktionen ein langer Weg weg von der eigentlichen Funktion.
    - dplyr ermöglicht es , Befehlen mit der Kette oder %.% Funktionen der Operationen linear zu sequenzieren, und damit viel mehr logisch.

- Datenbank

  - In Anlehnung an Hadley, unterstützt dplyr die drei populärsten Open-Source-Datenbanken (SQLite, MySQL und PostgreSQL) und Google BigQuery.

  - Nutzlich wenn wir unsere Daten aus der Datenbank nicht extrahieren wollen.

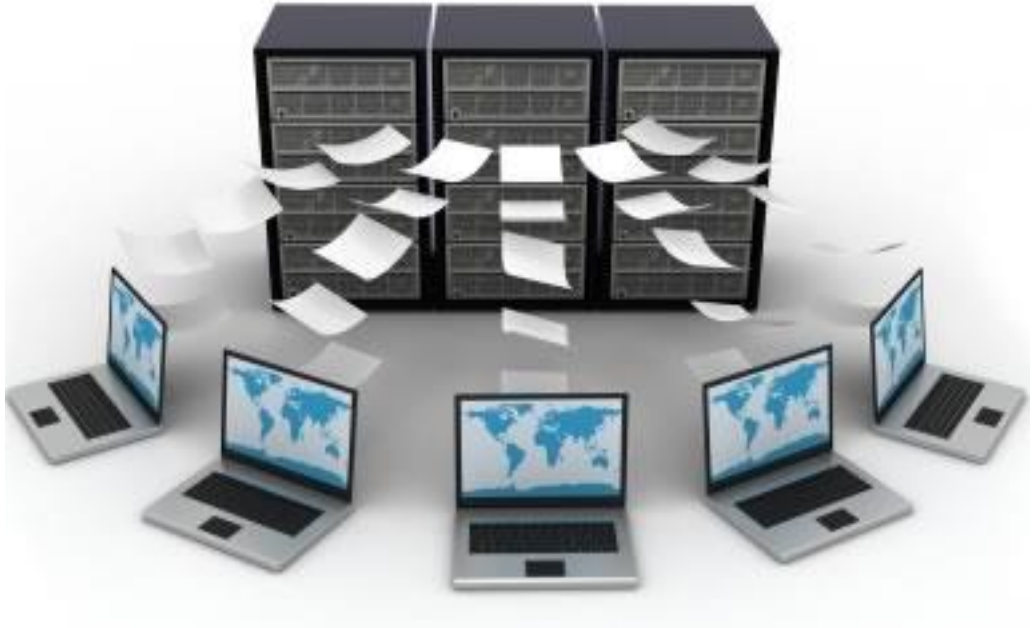

Abbildung 05
Quelle: „HardcoreAllainz“ [Ab05]

- Was ist reshape2
  - Ein R Paket die vereinfacht Dateisumwandlungsvorgang zwischen long und wide Format.
  - Ein Neustart des reshape Paket
  - Datensumformung deutlich stärker konzentiert und viel schneller
  - verbessert die Geschwindigkeit auf das Kosten der Funktionalität

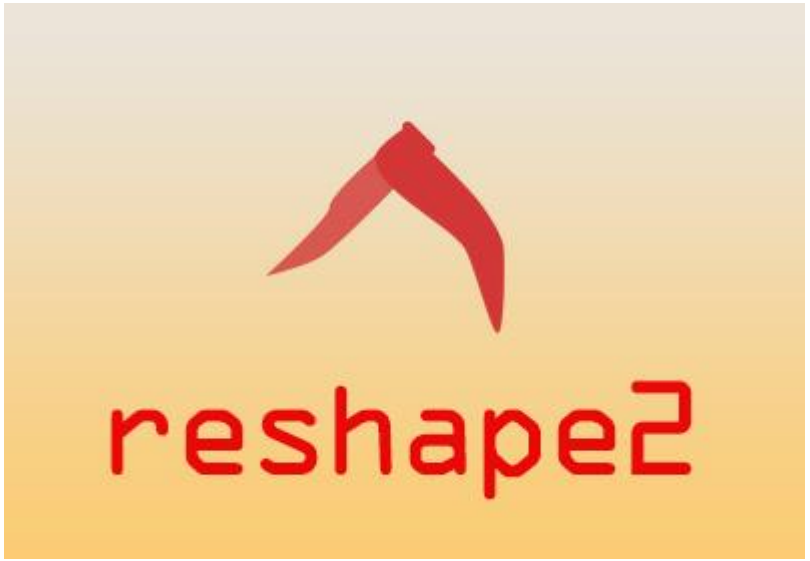

Abbildung 06
Quelle: „datascience+“ [Ab06]

- Verbesserung
  - wesentlich schneller und mehr Speicher effizient
  - cast wird durch zwei Funktionen in Abhängigkeit vom Ausgangstyp ersetzt: dcast erzeugt dataframe und acast erzeugt Matrizen / Arrays.
  - mehrdimensionale Margen sind jetzt möglich
  - Einige Funktionen wurden wie die entfernt zb. der| cast-Operator, und die Möglichkeit, mehrere Werte von einer Aggregationsfunktion zurückzuliefern.
  - eine bessere Entwicklungspraktiken wie Namespaces und Tests.

- Wide Dateisformat
  - Widesdatei hat eine Spalte für jede Variable
  - zb

```
#   ozone   wind  temp
# 1 23.62 11.623 65.55
# 2 29.44 10.267 79.10
# 3 59.12  8.942 83.90
# 4 59.96  8.794 83.97
```

- Long Dateisformat

```
#      variable  value
# 1    ozone 23.615
# 2    ozone 29.444
# 3    ozone 59.115
# 4    ozone 59.962
# 5     wind 11.623
# 6     wind 10.267
# 7     wind  8.942
# 8     wind  8.794
# 9     temp 65.548
# 10    temp 79.100
# 11    temp 83.903
# 12    temp 83.968
```

Beispiel 08
Quelle:“rdocumentation.org“ [Bsp08]

- Die Paket
  - reshape2 hat 2 basis Funktionen :
    - melt umwandelt wide -format nach long-format
    - cast umwandelt long -format nach wide – format
    - “Think of working with metal: if you melt metal, it drips and becomes long. If you cast it into a mould, it becomes wide.”

      -Sean Anderson [Zi01]

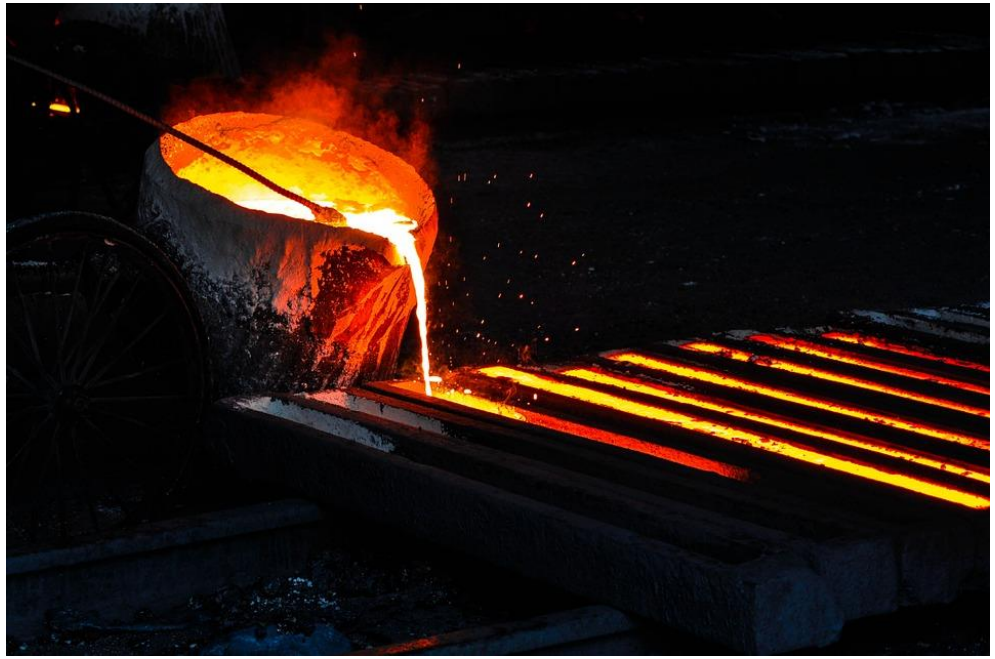

Abbildung 07

Quelle: „pixabay.com“ [Ab07]

- Zusammenfassung:
  - plyr
    - Ist ein R Paket das vereinfacht den Dateisaufteilungsvorgang, Dateisbearbeitungsvorgang und Dateisvereinigungsvorgang.
    - Die Basisformatte ist 2 Buchstaben und dann ply().
    - Ist nicht immer schnell
  - dplyr
    - ist die nächste Iteration von PLYR
    - Ziel ist nur Data frames
    - ist schneller als PLYR, hat mehr konsistent API
    - Manipuliertsfunktionen: filter, arrange, mutate, select, summarize
    - Channing command: dplyr ermöglicht es , Befehlen mit der Kette oder %.% Funktionen der Operationen linear zu sequenzieren, und damit viel mehr logisch als Nested R-Befehle.
  - reshape2
    - Ein R Paket die vereinfacht Dateisumwandlungsvorgang zwischen long und wide Format.
    - Ein Neustart des reshape Paket
    - Basisfunktionen:
      - melt umwandelt wide -format nach long-format
      - cast umwandelt long -format nach wide – format

- Literatur

  - https://cran.r-project.org
  - https://inside-r.org
  - https://rdocumentation.org
  - https://r-bloggers.com
  - The Comprehensive R Archive Network

Fragen?

Abbildung 04
Quelle:“textbroker“ [Ab04]

Vielen Dank für eure Aufmerksamkeit