P05 – dplyr, tidyr, Umgebungen

10. Mai 2021

Contents

Hinweise zur Abgabe:

Erstelle pro Aufgabe eine R-Code-Datei und benenne diese nach dem Schema `P<Woche>-<Aufgabe>.R` also hier `P05-1.R` und `P05-2.R` Schreibe den Code zur Lösung einer Aufgabe in die jeweilige Datei (zu Abgabe von Aufgabe 3 bei Aufgabenstellung).

Es ist erlaubt (aber nicht verpflichtend) zu zweit abzugeben. Abgaben in Gruppen von drei oder mehr Personen sind nicht erlaubt. Diese Gruppierung gilt nur für die Abgabe der Programmierprobleme, nicht für die Live-Übungen.

Bei Abgaben zu zweit gibt nur eine der beiden Person ab. Dabei müssen in **jeder** abgegebenen Datei in der **ersten Zeile** als Kommentar **beide** Namen stehen also zB

```
# Ada Lovelace, Charles Babbage

1+1
# ...
```

Die Abgabe der einzelnen Dateien (kein Archiv wie `.zip`) erfolgt über Moodle im Element namens `P05`. Die Abgabe muss bis spätestens Sonntag, 16. Mai 2021, 23:59 erfolgen.

Sportfest 2 (12 Punkte)

In der Datei `P05-1-exerc.R` wird zu Beginn ein Dataset durch Simulation erzeugt.

```
sports
## # A tibble: 1,000 x 7
##    name      grade letter time100 time200 time400 time1000
##    <chr>     <int> <chr>    <dbl>   <dbl>   <dbl>    <dbl>
##  1 Jess          8 a         25.0    52.3   127.      372.
##  2 Maxie         8 b         22.2    46.1   107.      285.
##  3 Iona         11 d         19.3    43.8    97.4     260.
##  4 Leslie        6 d         30.4    67.1   135.      449.
##  5 Mordechai     5 d         31.4    72.0   177.      521.
##  6 Meir         11 a         21.9    46.0   101.      266.
##  7 Nannette      9 a         24.9    48.5   117.      348.
##  8 Jevon         9 b         22.1    41.7   104.      258.
##  9 Mae          11 a         23.6    46.4    97.1     279.
```

```
## 10 Paisley       7 c         26.5    66.1   129.      413.
## # ... with 990 more rows
```

```
requirements
```

```
## # A tibble: 11 x 3
##    level min100 min1000
##    <int>  <dbl>   <dbl>
##  1     1     43     500
##  2     2     41     480
##  3     3     39     460
##  4     4     37     440
##  5     5     35     420
##  6     6     33     400
##  7     7     31     380
##  8     8     29     360
##  9     9     27     340
## 10    10     25     320
## 11    11     23     300
```

Die Tabelle `sports` enthält Zeiten des 100m-, 200m-, 400m- und 1000m-Laufs des Sportfests einer Schule (alles fiktiv). Die Spalten sind

- `name`: Vorname des Schülers / der Schülerin
- `grade`: Jahrgangsstufe (zwischen 5 und 11)
- `letter`: in jeder Jahrgangsstufe gibt es 4 Klassen: `a` bis `d`.
- `timeXXX`: die Zeit in Sekunden, die die Person zum Laufen der Distanz `XXX` benötigt hat

`requirements` gibt für 100 und 1000m und jede Jahrgangsstufe die geforderten Zeiten zum Bestehen an.

In der Datei `P05-1-exerc.R` stehen im unteren Teil Kommentar-Abschnitte markiert mit `a)` bis `c)`. Diese beschreiben eine erwartete Ausgabe, die mit `dplyr`-Verben erzeugt werden soll.

Schreibe einen Ausdruck, der die geforderten Werte liefert direkt unter den entsprechenden Kommentar.

Nutze ggf den Pipe-Operator `%>%` (Shortcut: `Ctrl` + `Shift` + `M` bzw `Cmd` + `Shift` + `M`).

Sofern nicht anders angegeben, müssen Spalten nicht extra ausgewählt werden.

Babynames (28 Punkte)

Das Tibble `babynames::babynames` enthält für jedes Jahr zwischen 1880 und 2017 getrennt nach Geschlecht Namen, die in einem Jahr mindestens 5-mal vergeben wurden (nach offizieller Statistik der *United States Social Security Administration*).

```
babynames::babynames
```

```
## # A tibble: 1,924,665 x 5
##     year sex   name          n   prop
##    <dbl> <chr> <chr>     <int>  <dbl>
##  1  1880 F     Mary       7065 0.0724
##  2  1880 F     Anna       2604 0.0267
##  3  1880 F     Emma       2003 0.0205
##  4  1880 F     Elizabeth  1939 0.0199
##  5  1880 F     Minnie     1746 0.0179
##  6  1880 F     Margaret   1578 0.0162
##  7  1880 F     Ida        1472 0.0151
##  8  1880 F     Alice      1414 0.0145
##  9  1880 F     Bertha     1320 0.0135
```

```
## 10  1880 F     Sarah      1288 0.0132
## # ... with 1,924,655 more rows
```

1. Erstelle eine neue Tabelle `bn20`, die das Vorkommen von Namen, getrennt nach Geschlecht, für die Jahre ab (einschließlich) 2000 enthält. `bn20` enthält 3 Spalten: `name`, `sex`, `n`.
2. Wir möchten Namen finden, die häufig sind sowohl für Mädchen als auch für Jungen vergeben werden. Erzeuge aus `bn20` eine Tabelle `bn20_sim` mit drei Spalten `name`, `female`, `male`. Die letzten beiden Spalten enthalten die Anzahl wie oft der Name ab dem Jahr 2000 an Mädchen oder Jungen vergeben wurde. In `bn20_sim` sollen nur solche Namen vorkommen, die sowohl an Mädchen als auch an Jungen und insgesamt (Mädchen und Jungen) mindestens 10^5-mal vergeben wurden. Ordne die Zeilen nach der relativen Differenz aufsteigend, dh $|n_{\text{male}} - n_{\text{female}}|/(n_{\text{male}} + n_{\text{female}})$. **Hinweis:** `bn20_sim` hat 117 Zeilen.
3. Erzeuge aus `bn20` eine Tabelle mit 2 Spalten `name_length`, `n`, die zählt wie oft Namen einer bestimmten Länge `name_length` vergeben wurden (unabhängig vom Geschlecht). Die Tabelle enthält keine Angaben mit dem Wert 0 bei `n` und alle Namen in `bn20` werden gezählt. **Hinweis:** Ergibt 14 Zeilen.
4. Zeige die Zeilen aus `bn20`, die einen Namen der kürzesten Länge enthalten, sortiert nach Häufigkeit. **Hinweis:** Ergibt 149 Zeilen.
5. Erzeuge aus `babynames` eine Tabelle `bn_avg_len`, die drei Spalten `year`, `sex`, `avg_len` enthält und für jedes Jahr und Geschlecht die durchschnittliche Länge des Namens enthält. **Hinweis:** `P05-1-exerc.R` enthält einen Aufruf von `ggplot2::ggplot()`, der das Ergebnis visualisiert.
6. Erzeuge aus `bn` angegeben in `P05-1-exerc.R` eine Tabelle `bn_trending` mit 6 Spalten `year`, `sex`, `name`, `n`, `n_prev`, `s_incr`, die die Top3 "trending" Namen der Jahre 2001 bis 2017 enthalten. Füge zuerst eine Spalte `n_prev` zu `bn` hinzu, die die Anzahl `n` aus dem Vorjahr enthält (außer für das Jahr 2000, dessen Vorjahr nicht mehr in `bn` enthalten ist). Hinweis: Ein Mutating-Join kann hier helfen. `s_incr` ist das geglättete relative Wachstum: $(n - n_{\text{prev}})/(n_{\text{prev}} + c)$, wobei hier c den Wert `mean(bn$n)` habe. Wähle für jedes Jahr und Geschlecht die 3 Namen mit dem höchsten `s_incr`-Wert aus.

Umgebungen (20 Punkte)

In dieser Aufgabe sollen Umgebungsdiagramme gezeichnet werden. Sie richten sich nach den Umgebungsdiagrammen der Vorlesung. Farben müssen aber nicht benutzt werden. Die Diagramme können per beliebigem Zeichenprogramm (Vorlage für LibreOffice Draw im Ordner `P05`) oder per Hand und Scan oder Foto erzeugt werden. Die Abgabe ist ein PDF oder eine gängie Bilddatei (PNG, jpeg). Gib eine (`P05-3.<Endung>`) oder zwei Dateien (`P05-3-1.<Endung>`, `P05-3-2.<Endung>`) ab.

Evil Parent

Die Funktion `env_parent()` aus dem Pakte `rlang` ist nicht als Ersetzungsfunktion vorhanden, Das Base-R Äquivalent `parent.env()` schon. Damit können wir nun üble Dinge tun...

```
library(rlang)
f <- function() {
  # hier
  x
}
e1 <- env()
fn_env(f) <- e1
e2 <- env(e1)
parent.env(e1) <- e2 # evil
f()
```

Zeichne ein Umgebungsdiagramm zum Zeitpunkt der Ausführung `f()` in der letzten Zeile, also bei der Markierung `# hier`. Die globale Umgebung soll gezeichnet werden, Umgebungen von Paketen nicht.

Call and Create

Der folgende Code entspricht in etwa dem letzten Beispiel aus der Vorlesung `V09-part1` (Ende von Abschnitt 6 in den lecture notes `NV09`).

```
x <- 4
y <- 4
z <- 4
crea <- function() {
  y <- 2
  function() {
    x <- 1
    # hier
    c(x=x, y=y, z=z)
  }
}
call <- function(g) {
  x <- 3
  y <- 3
  z <- 3
  print(g())
}
f <- crea()
z <- 5
call(f)
## x y z
## 1 2 5
```

Zeichne ein Umgebungsdiagramm, welches des Zustand der Programms an der mit `# hier` markierten Stelle (beim Funktionsaufruf der letzten Zeile) abbildet. Die globale Umgebung soll gezeichnet werden, Umgebungen von Paketen nicht.