

Anleitung für das Dienstplanprogramm

Anwendungsszenario/Was macht das Programm?

Das Programm erstellt einen Dienstplan für eine Gruppe von Ärzten, die die Nachtdienste eines Monats unter einander aufteilen müssen (z.B: alle Assistenzärzte einer Abteilung). Dabei berücksichtigt es allgemeine Präferenzen (z.B. bevorzugt jemand 25h-Nachtdienste oder 12.5h geteilt?), sowie tageweise Einschränkungen (z.B. “kann am 2.2. keinen Nachtdienst machen”). Das Programm versucht dann einen “optimalen” Dienstplan zu finden, bei dem die Dienste/Wochenenden/etc ausgewogen verteilt sind, und die resultierende Tagespräsenz ausgeglichen ist. Hier ein Beispiel für die tageweisen Einschränkungen:

2019-02	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
McCoy	!N?	!N	!N	U	U	U	U	U	!N	!N			!N1	!N				!N		!N1							!N1	
Crusher			N?				N?											!N										
Pulaski			!N	U					!N	!N	U	U	U	U	U	!N	!N		!N1									
Bashir	!N	!N						!N	!N	!N	U	U	U	U	U	!N	!N	!N										
TheDoctor	U	!N	!N	!N?		!N											!N	!N	!N		N?					!N		
Phlox	!N	!N	!N	U	U	U	U	U	!N	!N																		
Culber	X	!N	!N	U	U	U	U	U	!N	!N	N2?								!N			!N	!N	!N			!N	
Franklin	!N1																										!N	
Cottle	!N	!N	!N											!N	!N	!N	!N	U	U	U	U	U	!N	!N	U	U	U	U

!N	Kein Nachtdienst (auch kein N1 oder N2)
!N1	Kein N1 (auch kein N, aber N2 möglich)
!N2	Kein N2 (auch kein N, aber N1 möglich)
U,ZA,NG	Urlaub/Zeitausgleich/Nachtdienstgustunden
5,6,7,8	Normaler 5/6/7/8h-Tag
!N?	Kein N gewünscht (aber möglich)
N?	Nachtdienst gewünscht (aber nicht garantiert da es die Planung stark einschränkt)
N	Nachtdienst fix geplant - nur für Weihnachten o.Ä. vorgesehen
...	

...und ein daraus resultierender Dienstplan (mit den resultierenden Stationspräsenzen):

2019-02	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
McCoy	7			U	U	U	U	U			N1	7	7	7	FT	N	X	7	N2	X	5	5			5	N1	5	N2
Crusher	N	X		N	X	5	N	X			5	5	5	5	N	X		5	5	5	-	5			5	5	5	5
Pulaski	FT	N	X	U	7	N	X	7			U	U	U	U	U			5	5	5	5	5			5	N2	X	7
Bashir	5		N	X	5	5	5	5			U	U	U	U	U			5	5	-	5	N	X		N2	X	5	N1
TheDoctor	U			5	N	X	5	FT	N	X	5	5	5	5	5			5	5	5	N	X			N1	5	7	5
Phlox	5			U	U	U	U	U			5	5	5	5	5			N	X	5	5	FT	N	X	5	5	N	X
Culber	X			U	U	U	U	U			N2	X	5	N1	5		N	X	5	N	X	5			5	5	5	5
Franklin	5			5	5	5	5	N	X		5	N	X	N2	X			5	N1	5	5	5		N	X	5	5	5
Cottle	5			5	5	5	5	5		N	X	5	N	X	5			U	U	U	U	U			U	U	U	U

4_2_4_1	0.4	-1	-0.6	0	-1	-0.6					0	0.4	0.4	0.4	-1			1.4	0	0	0	1			1	0	0	0.4
4_3	0		0	1	0	1	0				0	0	0	0	0			1	1	0	1	0			0	0	1.4	1
12_2_6_3	0	-1	-1	-1	-1	-1					-1	0	0	-1	0			-1	-1	0	-1	-1			-1	0	0	-1
	0.4	-2	-0.6	-1	-1	-1.6					-1	0.4	0.4	-0.6	-1			1.4	0	0	0	0			0	0	1.4	0.4

Das Programm probiert dabei tausende Dienstpläne durch, und nimmt dann den “besten”. Welcher der beste ist, hängt von einigen Qualitätsparametern ab, deren Gewichtung man auch ändern kann (siehe später).

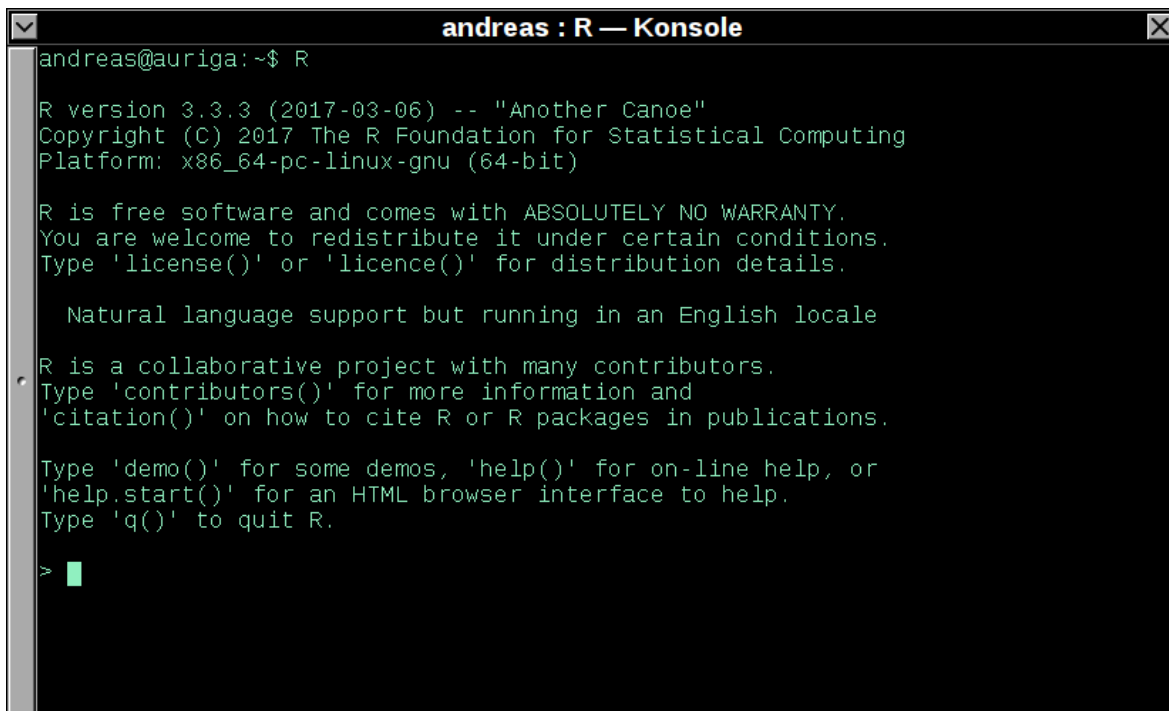
Vorbereitung

1 R installieren

Das Dienstplanprogramm ist in *R* geschrieben, eine Mathematik/Statistik-Programmiersprache. Diese muss zuerst installiert werden, gibts für alle gängigen (Linux, Windows, OSX) Betriebssysteme. Gehe zu <https://www.r-project.org/> und folge den Anleitungen. Falls bei der Installation gefragt wird, ist die 64bit-Version gegenüber der 32bit-Version zu bevorzugen.

2 Notwendige Pakete in R installieren

Es gibt zahlreiche ‘Pakete’ mit Zusatzfunktionen in *R*, das Dienstplanprogramm benutzt eins davon, *xlsx* (enthält Funktionen zum formatieren von Excel-Tabellen, die das Programm für Eingabe/Ausgabe benutzt). Dazu starten wir *R*: In Linux und OSX kann man einfach einen Terminal (“Kommandozeile”) öffnen und ‘**R**’ eingeben (dann Enter). In Windows bringt *R* einen eigenen Terminal mit, den findet man unter *Startmenü/Programme/R/etc.* (In OSX bringt *R* auch einen eigenen Terminal mit, *Rgui.app*, kann man alternativ auch verwenden). Wenn *R* läuft sieht das dann in etwa so aus:



```
andreas@auriga:~$ R
R version 3.3.3 (2017-03-06) -- "Another Canoe"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> █
```

Um das notwendige Paket zu installieren, folgendes im R-Terminal eingeben:

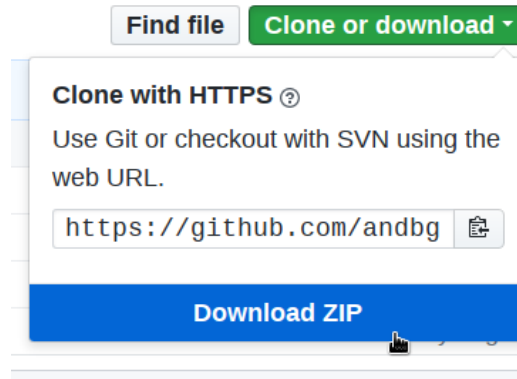
`install.packages("xlsx")` (dann Enter)

Eventuell wird man benachrichtigt, dass man noch eine Java-Bibliothek nachinstallieren muss, da *xlsx*

seinerseits davon abhängig ist, die muss dann auch noch installiert werden. Dabei muss die gleiche Prozessorarchitektur (64bit) wie bei R installiert werden, findet man unter <https://www.java.com/en/download/manual.jsp>, dort dann 64bit auswählen.

3 Dienstplanprogramm herunterladen und entpacken

Gehe zu https://github.com/andbgr/doctor_scheduling und lade das Dienstplanprogramm herunter:



Danach den ZIP Ordner irgendwo entpacken.

Anwendung

1 Neuer Ordner für neuen Monat erstellen

Geh in den Ordner *doctor_scheduling*. Es empfiehlt sich, für jeden Monat einen neuen Unterordner anzulegen. Der Ordner *example* dient als Vorlage, den kopieren wir, der neue Ordner heisst z.B. 2019-03, weil wir März machen.

2 Liste der Ärzte und allgemeine Präferenzen: *doctors.xlsx*

Öffne die Datei *doctors.xlsx* und gib die Ärzte mit ihren allgemeinen Präferenzen ein. Kopiere die Spalten *shifts_carryover* und *weekends_carryover* aus der *doctors.xlsx* des Vormonats, falls diese berücksichtigt werden sollen:

ward	Station
weekhours_min	Wochenstunden, normalerweise 40
weekhours_max	Wochenstunden maximal im Plan, normalerweise zwischen 40 und 48
split_shifts	‘yes’ oder ‘no’, macht der Arzt geteilte (12.5h) Dienste?
friday_sunday	‘yes’ oder ‘no’, macht der Arzt Freitag-Sonntag-Kombinationen?
keep_weekends_apart	Numerischer Wert von 0-6. Beschreibt, wie weit Dienste an konsekutiven Wochenenden auseinander liegen müssen. 0 bedeutet keine Einschränkungen, dann sind z.B. 2 konsekutive Samstage möglich (eher unerwünscht). 6 bedeutet, dass gar keine Dienste an 2 konsekutiven Wochenenden vergeben werden, nicht einmal Freitag...Sonntag. 1-5 sind Abstufungen dazwischen, 3 erlaubt z.B. Sonntag...Freitag, aber nicht Samstag...Samstag. Hinweis: höher ist natürlich wünschenswert, aber wenn ein Arzt z.B. nur 2 aufeinanderfolgende Wochenenden Dienst machen kann, dann macht es sinn, für diesen Arzt den Wert herunterzusetzen, da sonst wiederum die faire Verteilung der Wochenenden leidet.
number_of_shifts_factor	Faktor für die Anzahl an Diensten, normalerweise 1. Beeinflusst natürlich auch die anderen Ärzte. Beispiel für sinnvolle Anwendung: Neuanfänger macht noch nicht so viele Dienste (Faktor 0.5). Oder: es ist abgesprochen, dass 1 Arzt mehr Dienste macht (Faktor 1.2), und ein anderer weniger (Faktor 0.8).
min_day_hours	Normalerweise 5. Beschreibt die Mindeststunden an einem ‘normalen’ Tagdienst.
max_day_hours	Größer als min_day_hours, z.B. 7 oder 8. Beschreibt die Stunden an einem längeren Tagdienst, falls mit kurzen Tagdiensten die Sollstunden noch nicht erreicht wurden.
preferred_day_hours	‘min’ oder ‘max’. Beschreibt, ob bis zum Erreichen der Sollstunden kurze Dienste an allen Tagen vergeben werden, oder (sofern es die Tagespräsenz erlaubt) mehr längere Dienste und mehr 0-h-Tag (“Stricherltag”).
fill_all_days	‘yes’ oder ‘no’, Wenn die Sollpräsenz der Station UND die 40 Wochenstunden des Arztes erreicht sind, sollen dann trotzdem noch mehr Tagdienste vergeben werden? (Wenn nicht: Stricherltag). Die Einschränkung durch weekhours_max gilt aber trotzdem.
shifts_carryover	Wenn im letzten Monat ‘zu viele’ oder ‘zu wenige’ Dienste (gemessen an einem rechnerischen Soll, mit Kommastellen) gemacht wurden, kann man die aus der Statistik

	des Vormonats hierher kopieren, dann wird das berücksichtigt. Achtung: beim Kopieren auf die Reihenfolge der Ärzte achten, falls sich diese zum Vormonat verändert hat.
weekends_carryover	Wie oben, nur für Wochenenden

3 Dienstplanprogramm ausführen – neue *input.xlsx* schreiben

In unserem Ordner liegt 2 Helferskripte, *1-write_template.R* und *2-create_schedule.R*, von denen aus die Funktionen aus dem eigentlichen Dienstplanprogramm (*doctor_scheduling.R*) aufgerufen werden. Wir verwenden zuerst *1-write_template.R*, um eine neue neue *input.xlsx* Tabelle zu schreiben:

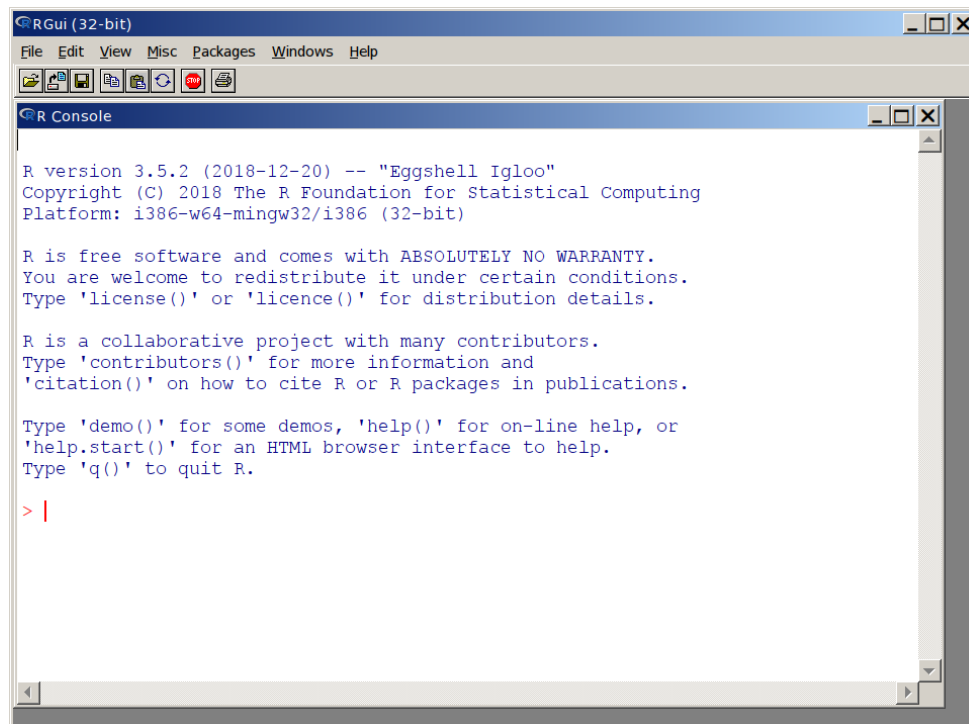
Linux/Allgemeine Anleitung

1. Öffne *1-write_template.R* im Texteditor deiner Wahl und passe `start_date` und `end_date` an, speichere die Änderungen.
2. Öffne einen Terminal in unserem Ordner und führe das Skript aus:

```
Rscript 1-write_template.R
```

Windows/RGui

1. Starte *R* (*Startmenü/Programme/R/etc*). Das sieht dann in etwa so aus:



2. Wechsle das Arbeitsverzeichnis, gehe zu unserem Ordner (*2019-03*) mit Menü **File > Change**

dir...

3. Öffne 1-write-template.R mit File > Open Script...
4. Passe start_date und end_date an
5. Führe das Skript aus, mit Edit > Run all

4 Befülle input.xlsx

Oder: Stelle *input.xlsx* den Kollegen zur Verfügung, um ihre Wünsche einzugeben. Die Angabe für validen input findet sich in der Datei. Einige Dinge sollten nicht vergessen werden:

- trage das X am ersten Tag für den Kollegen ein, der am letzten Tag des Vormonats Dienst hatte
- trage !N am ersten Tag für den Kollegen ein, der am vorletzten Tag des Vormonats Dienst hatte, um einen Doppeldienst zu vermeiden.
- (optional) trage !N am ersten Wochenende für die Kollegen ein, die am letzten Wochenende des Vormonats dienst hatten (analog zu keep_weekends_apart, siehe oben), um so etwas wie Samstag-Samstag zu vermeiden.

5 Dienstplanprogramm ausführen – Dienstplan generieren

1. Analog zur Anleitung oben, öffne nun das Skript 2-create_schedule.R.
2. Passe n.iterations an. Diese Variable bestimmt, wie viele Dienstpläne durchprobiert werden. Je nachdem wie viele Einschränkungen es gibt, braucht es zwischen ca 7000 oder 70000. Dies dauert (Erfahrungswert: auf einem durchschnittlichen Laptop Baujahr 2015 gehen sich etwa 14000 Iterationen in 1h aus).
3. Passe die Gewichtungen (weights) der Optimierungsparameter an. Diese bestimmen, wie hoch die verschiedenen Optimierungsparameter gewichtet werden (höher = wichtiger, 0 = gar kein Einfluss). Zur Erklärung der Optimierungsparameter siehe unten. Probiere verschiedene Werte aus und vergleiche die resultierenden Dienstpläne!
4. Starte das Skript (analog zur Anleitung oben).
Während das Programm läuft, sieht das in etwa so aus:

```
Optimizing - trying 7000 variations...
n.unres n.rq_den n.srq_granted r.shifts r.weekends r.nights n.split day_presence
0      0      2/7      1.306      0.8444      3      9/16      24.4 (28.9)
0      0      2/7      1.806      0.6556      1      8/16      24.6 (26.4)
0      0      5/7      1.667      0.8222      1      6/16      22.6 (22.3)
0      0      6/7      1.194      0.7222      1      6/16      23.6 (25.0)
0      0      6/7      1.028      0.8222      1      7/16      24.8 (26.1)
0      0      4/7      1      0.8222      1      8/16      21.8 (21.0)
0      0      6/7      1.25      0.4556      1      8/16      23.4 (24.8)
0      0      6/7      1.028      0.4556      1      8/16      25.8 (29.1)
0      0      5/7      0.7778      0.4778      1      9/16      23.0 (24.4)
```

Dabei repräsentiert jede Zeile einen neuen Dienstplan mit seinen Qualitätsparametern. Es wird aber nur eine neue Zeile ausgegeben, falls der neue Dienstplan besser als der vorige ist. Somit

geht es am Anfang schnell, dann kommen immer seltener neue Zeilen hinzu. Am Ende wird der 'beste' Dienstplan dann in die Datei *schedule.xlsx* geschrieben. Die Differenz zur Sollpräsenz je Station steht unter dem Dienstplan, ausführliche Statistik im Tabellenblatt *doctors.stats*. Bitte auch im Tabellenblatt *warnings* nachsehen, ob es Probleme gibt (z.B. ein FT konnte nicht vergeben werden o.Ä.). Hier noch eine Erklärung der Qualitätsparameter:

n.unres	Anzahl der ungelösten Tage (kein Nachtdienst) – sollte 0 sein (ausser es geht sich nicht aus)
n.rq_den	Anzahl der Verstösse gegen Einschränkungen – sollte 0 sein (ausser bei unauflösbaren Eingaben, z.B. fixe Eingabe von N bei 2 Ärzten am gleichen Tag – da wird dann einer rausgeworfen)
n.srq_granted	Anzahl der "soft requests" die erfüllt wurden. Das sind alle Eingaben mit einem '?'
r.shifts	Range, also Spanne der Anzahl der Nachtdienste. Ein Beispiel: Das rechnerische Soll an Nachtdiensten in einem Monat ist 3.6. Jetzt haben die Ärzte zwischen 3 und 4 Dienste, dann wäre r.shifts 1, weil maximale Abweichung nach unten (0.6) plus maximale Abweichung nach oben (0.4) = 1.
r.weekends	Wie oben, nur für Wochenenden. Aktuell zählt hier ein Samstag als 1, Freitag 0.4, Sonntag 0.6 (im Sinne von 'betroffene Wochenenden', als Dienste werden sie natürlich separat gezählt).
r.nights	Maximaler Unterschied Anzahl N1 vs N2 bei einem Arzt – sollte 0 oder 1 sein
n.split	Wie viele der prinzipiell teilbaren Dienste (Alle Mo-Do) wurden geteilt?
day_presence	Anzahl der fehlenden Ärzte-Tage gesamt. Die Zahl in (Klammer) ist das gleiche, aber quadriert bevor es addiert wurde. Ein Beispiel: Es fehlen jeweils 1 Arzt an 2 Tagen: $1+1=2(1^2+1^2=2)$ Es fehlen 2 Ärzte an 1 Tag: $2=2(2^2=4)$ Mit anderen Worten: wenn die Zahl in Klammer deutlich höher ist, dann fehlen irgendwo mehrere am gleichen Tag. Der Wert in Klammer wird bei der Optimierung verwendet, es wird also stärker bestraft, wenn an weniger Tagen mehrere fehlen als umgekehrt

Hinweise zur sinnvollen Anwendung:

Es ist wesentlich besser, einzugeben, wann jemand **nicht** Dienst machen kann (oder will), als einzugeben, wann jemand Dienst machen will. Zweiteres schränkt die Freiheitsgrade in der Planung stark ein, und führt zu einem weniger ausgewogenen Dienstplan.

Prinzipiell ist aber jeder Dienst auch als Eingabe möglich (z.B. "N", aber auch "FT" oder "-"). Wenn man den gesamten Dienstplan aus dem Ergebnis wieder als Wünsche in den Input kopiert, kommt der exakt gleiche Dienstplan wieder raus. Somit könnte man z.B. auch einen halb fertigen Dienstplan nehmen, und den Rest dem Programm überlassen. Oder Weihnachten fix verplanen, und den Rest offen lassen, etc

Hinweis: das Programm ist in Entwicklung und hat mit Sicherheit Fehler und Verbesserungsmöglichkeiten, es ist also sinnvoll ab und zu auf der GitHub-Seite nach Updates zu sehen:

https://github.com/andbgr/doctor_scheduling

Fragen oder Verbesserungsvorschläge an:

andi.berger@yandex.com