Übungsaufgabe R Tutorium CIPOM@Charité

**KONTEXT:** Für ein blutdrucksenkendes Medikament soll ein t.test() prüfen, ob das Medikament einen entsprechend signifikanten Unterschied macht, wenn man den sysRR vor und nach Gabe des Medikamentes miteinander vergleicht. (Bedenke, es handelt sich um eine stark vereinfachte Übungsaufgabe mit überschaubaren Datenset, daher bspw. nur sysRR vorhanden usw.).

Lese jeden Punkt einmal komplett, bevor du mit der Lösung beginnst.

1. Erstelle ein neues Script (File->New-File->New Script). Zunächst importiere das Datenset via read.csv(„file\_path\_and\_name“). Und wähle einen Objektnamen (quasi Variablen-/Bezugsname für die importierte Tabelle). ([Chapter 6](https://journal.medicine.berlinexchange.de/pub/4wdsmd0f/release/8#import-and-export-of-csv-files-understanding-delimiters-and-packages) bzgl. Import/Export) Prüfe, ob die Daten normalverteilt sind via shapiro.test() und plot(density()) – *je für die sysRR Werte für t1 und t2*. Du bekommst ggf. einen Fehler. Sehe dir für die Fehlerprüfung einmal die Spalte mit den sysRR in der Konsole an (gehe dann zu b) für die Lösung des Problems).
2. Bei Proband 5 fehlt der Eintrag für t1 in der Spalte sysRR. Trage den Wert „122“ mmHg sysRR nach, den du in deinen Unterlagen gefunden und vergessen hast, in deine Excel-Tabelle einzutragen... *Vorher ist ein Shapiro Wilck Test und ein t-test nicht möglich, aufgrund des NA Eintrags*. Erinnere dich daran, wie man bestimmte Positionen ansteuert, u.a., um Einträge zu ändern – ganz am Anfang als wir die c() Funktion besprochen haben. ([Chapter 5](https://journal.medicine.berlinexchange.de/pub/4wdsmd0f/release/8#classes-of-objects-vectors-matrices-arrays-lists-and-more) oder gucke direkt ins entsprechende Kapitel des R Basic Code Scriptes in diesem Falle)
3. Wie sich herausstellt, sind die Daten normalverteilt (ansonsten hätten wir einen Wilcoxon signed rank test verwenden müssen). Nun, welche Art von t-test wird benötigt? Vollziehe einen t-test via t.test() Funktion. Recherchiere dabei im Internet wie die Funktion benutzt wird bzw. siehe dir die Dokumentation via ?t.test() an und nutze auch das package effsize und die Funktion cohen.d()! Bedenkt bei der Auswertung auch die Mittelwerte, Mittelwert Differenz und Standardabweichung mit in die Betrachtung zu beziehen. Stellt euch vor, ihr habt eine sd von -0.1 mmHg, dann sind 2 sd standardisierte Mittelwertdifferenz (= effect size) wenig bzw. irrelevant (daher Angaben wie „large“ usw. durch die cohen.d() Funktion nicht unbedingt ernst nehmen sondern selber reflektieren; Skalierung stammt von Cohen selber)! Es gibt verschiedene Arten von SD für verschiedene t-test Formen. Erinnere dich, dass wenn du dem Output von bspw. t.test() einen Namen gibst, dann kannst du die einzelnen Werte, wie bspw. Mittelwertdifferenz (dort genannt estimate) oder sd aus einer Liste aufrufen!
4. In der Regel würde man eine Poweranalyse vor der Datengewinnung und Auswertung vollziehen. Welche sample size würdest du in der Regel für die obige Studie benötigen, gegeben eine Power von 80% und einem CI von 95%, sowie einer hypothetischen effect size von .5? Nutze hierfür die pwr.t.test() Funktion des packages pwr.
5. Räume die Spalte „fam“ = Familienanamnese auf. Ersetze die enthaltenen NA Einträge mit der Aussage „keine Angabe“. Nutze hierfür die Funktionen is.na(), which() and unique().([Chapter 8.6](https://journal.medicine.berlinexchange.de/pub/4wdsmd0f/release/8#example-vi-adjusting-character-string-entries))
6. Erstelle dann ein Kreisdiagramm mit der Funktion pie(table(tabelle\_name$fam)). Erstelle mit table() aus der fam Spalte eine Tabelle mit Häufigkeiten der einzelnen Kategorien. Reflektiere dabei die Struktur der Tabelle, die du für die Ermittlung der Häufigkeiten als Input wählst (Redundanzen vorhanden?).
7. Wenn alle Spalten aufgeräumt sind, exportiere die Tabelle als .csv Datei mit der write.csv(dein\_tabellen\_name, „Dein Dateipfad und Dateiname.csv“) Funktion. ([Chapter 6](https://journal.medicine.berlinexchange.de/pub/4wdsmd0f/release/8#import-and-export-of-csv-files-understanding-delimiters-and-packages) bzgl. Import/Export)

Bonusaufgaben

1. Erstelle zur Übung einen kleinen synthetischen Datensatz um die na.omit() Funktion oder eine andere Funktion zu testen. Bspw. zwei Spalten und 3 Zeilen, jede Spalte enthält jeweils ein NA. Nutze dafür die c(), cbind() und ggf. as.data.frame() Funktion. ([Chapter 8.5.1](https://journal.medicine.berlinexchange.de/pub/4wdsmd0f/release/8#representation-of-solutionalgorithm-ii-from-example-ii-via-tidyverse-and-the-dangers-of-the-naomit-function))
2. Reformatiere die Tabelle so, dass der sysRR von t1 und t2 nebeneinander in einer Tabelle sind. Die Redundanzen in der patient\_id und der Spalte fam verschwinden dadurch auch. Du kannst dafür die Funktionen cbind() verwenden. Ggf. brauchst du auch die Funktion colnames(). ([Chapter 5](https://journal.medicine.berlinexchange.de/pub/4wdsmd0f/release/8#classes-of-objects-vectors-matrices-arrays-lists-and-more) and [Chapter 8.13](https://journal.medicine.berlinexchange.de/pub/4wdsmd0f/release/8#example-xiii-converting-wide-into-long-format))
3. Füge eine weitere Spalte zu dem Datensatz hinzu und trage dort algorithmisch via R die sysRR Differenz ein. Nutze hierfür ebenfalls cbind() und c(0) für eine leere Spalte mit nur Nullen an jeder Stelle. ([Chapter 8.9](https://journal.medicine.berlinexchange.de/pub/4wdsmd0f/release/8#example-ix-working-with-redundant-columns))
4. Überprüfe via R wie viele Teilnehmende eine RR Differenz von größer/gleich 10 vorweisen. (Lösungsweg nur in den Solutions oder Internet, nicht in den Folien). Nutzer hierfür den Operator >= und die which() Funktion (bisher haben wir nur == verwendet).
5. Optional kann noch eine schöne Tabelle mit R erstellt werden. Nutze den folgenden Code dafür:

# install.packages("gt")

library(gt)

gt(reformatierte\_tabelle)

**Typische Fehler/Probleme beim Lösen der Aufgabe und im Umgang mit R generell**

**General Advice:** Versuche dich bei der Code Eingabe und beim Ausführen von Code gut zu konzentrieren, um generell Leichtsinnsfehler wie Typos etc. zu vermeiden. Sollte du verzweifelt sein, versuche dich zuvorderst daran zu erinnern, dass es eigentlich nur um (meist 2D) Tabellen geht, bei denen man irgendwas ansteuern will (Spalten, bestimmte Positionen in Spalten, ggf. unter bestimmten Kriterien usw.). Generell braucht man keine magischen Funktionen und auch keine tausend packages, wenn es nur um Tabellenaufräumen und auswerten geht. Wenige Funktionen reichen in der Regel aus. Ansonsten findest du unten ein paar allg. Fehler, die einem immer wieder mal passieren (nicht nur am Anfang oder speziell beim Lösen der Übungsaufgabe).

* Falscher Name/Buchstabendreher/Typo etc.
* Gucke ob Fehlermeldung durch RStudio angezeigt wird (roter Punkt mit weißem X), ggf. fehlt eine Klammer oder ein Komma zwischen den Parametern innerhalb der Klammern…
* Zeile noch nicht ausgeführt, nur Name für Anzeige in Konsole ausgeführt (Fehlermeldung in etwa: Object not found)
* + in der Konsole übersehen. Setze Klammer in leere Zeile und führe diese aus (lösche danach die Klammer wieder). Danach sollte wieder > in der Konsole am Ende erscheinen (signalisiert, dass Prozess abgeschlossen wurde). Geht + nicht weg, dann RStudio neustarten (Script und workspace kann gespeichert werden).
* Das Objekt t1 und t2 stammt noch von einem anderen Beispiel, daher auch ggf. andere Ergebnisse beim t-test als in den Lösungen vermerkt...
* Zeile ausgeführt, aber man hat vergessen, dass man Namen markieren und ausführen muss, um sich Ergebnis in Konsole anzeigen zu lassen (alternativ via print(objekt\_name))
* Ihr habt eine ganze Tabelle ausgewählt, nicht nur eine Spalte (bspw. bei t.test())
* Klasse ist nicht numerisch und muss geändert werden via as.numeric()
* Versuch in der Konsole zu coden, anstatt in einem Script…
* Import nicht möglich weil Projekt nicht geöffnet oder sonst wie verloren gegangen (entweder neues Projekt erstellen, oder altes öffnen), oder ganzen path eingeben! Nutze ggf. getwd() um aktuellen working directory zu erfahren…
* Immer Anführungszeichen bei character strings, ansonsten denkt R es handele sich um ein Objekt oder eine numeric Zahl bei Ziffern im Script…
* == für Äquivalenz, nicht nur ein „ist-gleich“ Zeichen!
* Package nicht geladen oder noch nicht installiert. Kann Grund sein dafür, dass die filter() Funktion nicht funktioniert…
* Unübersichtliches Codescript, sodass Fehler nicht leicht erkannt werden. Immer Kommentar vor jeder Zeile Code hinzufügen für Transparenz und Lesbarkeit.
* Objekt umbenannt, aber im Code steht noch der alte Name; beim Neustart bzw. nach Bereinigung des workspace/environment funktioniert dann der Code nicht mehr
* Code remnants aus dem Script löschen oder ans Ende des Scriptes, oder in ein Reste-Script packen!
* …
* …
* …