Praktická časť diplomovej práce

1 Sprievodca ekfmetriou

Sprievodca ekonometriou má za úlohu priblížiť Vám ekonometriu, a pomôcť Vám jej porozumieť. Sprievodcu píšem ako študent, ktorý sa ekonometriu začal učiť sám, a sám si prešiel zdĺhavým procesom bádania a usmerňovania. Sprievodca je zostrojený ako-tak súbežne s osnovou a zadaniami, ktoré obdržíte na hodine. Nebudeme sa konkrétne držať vypracovania zadaní, ale skôr princípmi, z ktorých zadania ťažia. Mnoho študentov tento predmet nezaujíma, a zadania vypracujú okopírovaním postupov starších spolužiakov, nuž, pochopiť ekonometriu a jej postupy nie je vôbec jednoduché, a dokážem pochopiť, keď si študenti hľadajú skratky. Na druhú stranu, ekonometria predstavuje skvelú vstupnú bránu do sveta analytiky. Človek je zavalený Machine Learningom, Data Sciencom a AI-čkom, nuž až po sfúknutí pozlátka zistí, že je to zmes matematiky, štatistiky a počítačovej vedy (CS – computer science). Ekonometria je teda skvelou výhovorkou, ako oprášiť matematiku, doučiť sa štatistiku, a naučiť sa troška programovania. R-ko sa môže zdať ako jazyk, ktorý žije v tieni Pythonu, avšak, akoby nejedna Dominika vedela povedať, netreba sa nechať voviesť do omylu. R-ko je najvhodnejší programovací jazyk pre štatistikov, Google ho zahrnul do najnovších kurzov Google Analytics. Mojou úlohou je pomôcť Vám prekonať problém, ktorý som na začiatku svojej cesty ekonometriou vôbec nepovažoval za problém, a to množstvo materiálov, ktoré zavalí študenta. Pomôžem Vám postupne zložiť skladačku konceptov a teórií, na ktorých ekonometria stojí. Náročnosť prezentovania konceptov bude prispôsobená. Nemá zmysel vysvetľovať odvodzovanie každého estimátora. Cieľom je poskytnúť všeobecný náhľad ekonometrie, a pomôcť Vám pochopiť, a teda príjemnejšie zvládnuť predmet Ekonometria.

Čo nás čaká:

- Základy programovania v R
- Intuitívny prehľad štatistických konceptov
- Ekonometrické techniky

2 Základy programovania v R

Sprievodca je interaktívny, teda začneme stiahnutím a inštaláciou R a RStudia. R má samo o sebe programovacie prostredie, avšak dnešným štandardom je používanie intregrovaného vývojového prostredia (IDE) v podobe RStudia.

2.1 Aritmetické operátory

Poďme teda rovno na vec. Začneme základnými funkciami. R môžeme používať ako kalkulačku, teda za pomoci klasických aritmetických operátorov môžeme sčítať, odčítať, násobiť, deliť či umocňovať:

```
5 + 5
```

```
## [1] 10
```

```
5 - 5
```

```
## [1] 0
```

```
5 * 5
```

```
## [1] 25
```

```
5 / 5
```

```
## [1] 1
```

```
5^2
```

```
## [1] 25
```

R-ko dokáže používať aj ďalšie aritmetické operátory:

```
# zobrazí zvyšok z delenia

5 %% 2
```

```
## [1] 1
```

My sa budeme zapodievať len tým, s čím sa na cvičeniach stretneme. Našou úlohou nie je naučiť sa dokonalo ovládať R, ale naučiť sa používať ho v dostatočnej miere, aby sme s ním zvládli to, čo budeme v najbližšej dobe potrebovať.

2.2 Balíky

Okrem základných operátorov budeme využívať aj funkcie:

```
mean(2, 4, 6)
```

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

```
abs(-5)
```

```
## [1] 5
```

```
sqrt(8)
```

```
## [1] 2.828427
```

Tieto funkcie sa nachádzajú v balíkoch, ktoré si môžeme predstaviť ako také Addony. R figuruje balíkmi, ktoré sú predinštalované, a zahŕňajú najpoužívanejšie a najzákladnejšie funkcie. Vyššie použité funkcie sa nachádzajú v balíku base. To, v akom balíku sa funkcia nachádza zistíte po napísaní funkcie

to však len zapredpokladu, že už máte balík nainštalovaný. Ak narazíte na názov funkcie, ktorú chcete použiť, avšak nemáte nainštalovaný balík a chcete zistiť jeho názov, buď si funkciu zadajte do Google, alebo napíšte do konzoly:

```
# ??meno funkcie

??mean
```

My budeme často využívať predinštalované balíky **base** a **stats**, avšak za pochodu si budeme inštalovať aj ďalšie balíky, s ktorými sa na cvičeniach stretnete. Nový balík je potrebné prv nainštalovať a potom ho načítať do prostredia.

```
# stiahneme a našintalujeme pomocou R konzoly a funkcie install.packages()
# ! názov balíka je citlivý na veľkosť písma
# ! názov musí byť v úvodzovkách

install.packages("fBasics")

# po nainštalovaní máme balík stiahnutý v našom PC, a tento príkaz už viac nepoužívame
# ak však chceme funkcie z balíka použiť, musíme po zapnutí R-ka balík načítať príkazom

library(fBasics)

# ! tu už úvodzovky nie sú potrebné
```

> *Pri inštalovaní názov balíka zabalíme do úvodzoviek. Pri jeho načítaní pomocou library() už úvodzovky nepíšeme. Na pohovor si vezmeme oblek (úvodzovky), ale po prijatí už chodíme do práce bez obleku.*

2.3 Objekty

Často chceme vyrátané výsledky znova použiť, a preto by bolo vhodné si ich niekde uložiť. Na ukladanie a uskladnenie výsledkov slúžia objekty. Každý objekt má meno a obsah. Meno si môžeme zadať akékoľvek, musí však:

- začínať malým alebo veľkým písmenom a nie číslom
- obsahovať iba čísla, písmená alebo niektoré špeciálne znaky ako "." či "_".

Nezabúdajme, že R je case sensitive (rozlišuje veľké a malé písmená).

Povedzme že chceme vytvoriť objekt **a** a priradiť mu hodnotu $2 + 2$. Na priradenie obsahu je možné použiť " = ", avšak štandardom je používanie " < − ".

*Znak <- nie je nutné písať dvoma znakmi, používa sa na to skratka **"ľavý Alt" a "-"**. Na SK klávesnici nájdeme znak naľavo od pravého Shiftu, na EN klávesnici zvyčajne naľavo od Backspacu. Osobne programujem s EN klávesnicou, nech som použíteľný v akomkoľvek štáte bez potreby inštalovať SK klávesnicu.*

```
# Priradíme teda objektu "a" výsledok "2 + 2".

a <- 2 + 2

# Po napísaní názvu objektu do konzoly nám konzola ukáže už len výsledok.

a
```

```
## [1] 4
```

```
# Nové priradenie hodnoty starému objektu prepíše starú hodnotu.

a <- 5 + 5

a
```

```
## [1] 10
```

S daným objektom môžeme pracovať, akoby to bola číselná hodnota.

```
a^2
```

```
## [1] 100
```

2.4 Vektory

Pre priradenie viacerých hodnôt vytvoríme vektor. Vektor vytvoríme pomocou funkcie c() ako combine.

```
# Objektu "a" priradíme súbor číselných hodnôt a vytvoríme z neho vektor.

a <- c(5, 1, 4, 2, 3)

a
```

```
## [1] 5 1 4 2 3
```

S vektormi dokážeme rôzne pracovať. Sčítavať, násobiť, aplikovať na ne funkcie, všetko, čo nás napadne. Ups, čo nám napadne!

```
b <- c(5, 9, 6, 8, 7)
a + b
```

```
## [1] 10 10 10 10 10
```

```
min(a)
```

```
## [1] 1
```

```
sort(a)
```

```
## [1] 1 2 3 4 5
```

```
sum(a)
```

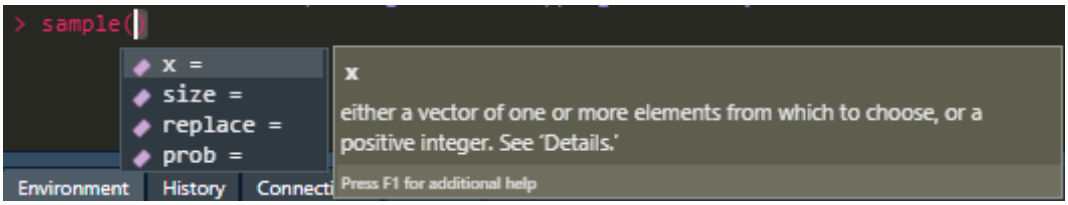
```
## [1] 15
```

Je vhodné poznať pár špeciálnych funkcií na tvorbu vektorov. A to:

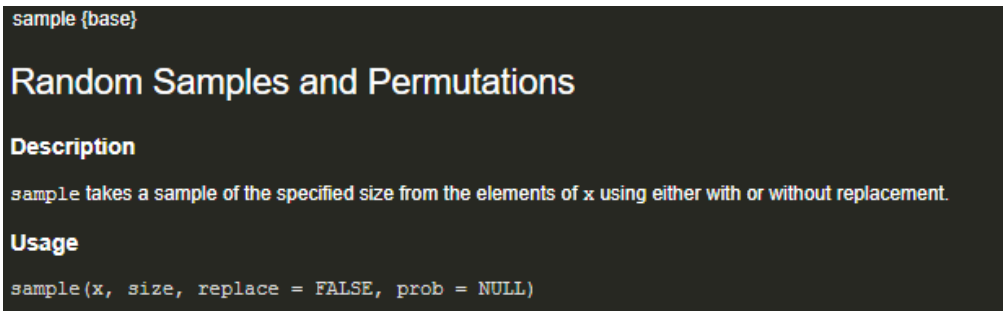
```
# sample(), seq() a rep()
```

Často si totižto potrebujete vytvoriť vektor hodnôt podľa vlastnej potreby. Hm, to som veľmi nič nové nepovedal. Ale... Čo vlastne tieto funkcie robia? A ako to zistíme? Ak pracujete s úplne novou funkciou, a máte už nainštalovaný balík, a taktiež ste ho už načítali do knižnice cez *library*(), je vhodné použiť ?*názovfunkcie* alebo *help*(*názovfunkcie*). Poskytne Vám to stručný popis, čo funkcia robí, všetky jej argumeny a aj pár príkladov. Ak už ste však s funkciou pracovali, ale nepamätáte si argumenty, rozpíšte funkciu, kliknite sa do zátvoriek funkcie *mean*(*tu.sa.kliknete*), a stlačíte *TAB*. Vyhodí Vám to argumenty funkcie. Skúsime si to na funkcii *sample*().

2.4.1 sample()



Pri jednej z vecí, ktorú Vám chcem ukázať, budeme potrebovať vektor výšky ľudí. Tak si ho teda vytvorme už teraz. Ešte predtým môžeme skúsiť napísať do konzoly ?*sample*, čím zistíme, že funkcia *sample*() nám náhodne vyberie hodnoty zo zadaného vektora.




```
# Ako prvé potrebujeme zadať "x", teda vektor hodnôt, z ktorých chceme vyberať.
# Pre nás to budú hodnoty medzi 160 až 190, ktoré navolíme ako 160:190.

# 160:190 jednoducho vypíše hodnoty zaradom od 160 do 190. A sample() z nich náhodne vyberie
# nami určený počet hodnôt.


# "size" je počet hodnôt, aký ma funkcia vybrať.

# "replace", či môže vybrať jednu hodnotu viackrát, alebo ju má vylúčiť.

# "prob" použijeme, ak chceme priradiť istým hodnotám inú váhu.

vyska <- sample(x = 160:190, size = 10, replace = TRUE)

print(vyska)
```

```
##  [1] 187 177 164 160 172 180 168 161 178 168
```

> *Funkcia by fungovala, aj keby sme to napísali ako* "sample(160:190, 10, TRUE)". *Je však vhodné písať aj argumenty. Hlavne pri funkciách, ktoré nie sú veľmi bežné. Ak po Vás niekto bude čítať kód, číta sa to lepšie.*

2.4.2 seq()

Funkcia *seq*() vygeneruje sekvenciu čísel. Ako argumenty zadávame buď *od*, *do* a *by*. Teda po akých inkrementoch bude daná sekvencia narastať.

```
seq(from = 2, to = 12, by = 0.5)
```

```
##  [1]  2.0  2.5  3.0  3.5  4.0  4.5  5.0  5.5  6.0  6.5  7.0  7.5  8.0  8.5  9.0
## [16]  9.5 10.0 10.5 11.0 11.5 12.0
```

> *V Rku používame na oddeľovanie desatinných miest bodku. Čiarkou oddeľujeme argumenty. Taktiež nezabúdajte používať medzery. Jedná sa o gramatiku programovania. Predsalen náš vyššie napísaný výraz vyzerá lepšie ako* seq(from=2,to=12,by=0.5).

```
# Zadáme "od", "do" a koľko častí má vektor mať.

seq(from = 4, to = 10, length = 4)
```

```
## [1]  4  6  8 10
```

```
seq(from = 4, to = 10, length = 8)
```

```
## [1]  4.000000  4.857143  5.714286  6.571429  7.428571  8.285714  9.142857
## [8] 10.000000
```

```
# Alternatívou je zadať "od", "po", a rozdeliť to po požadovaných kúskoch
# na požadovanú dĺžku.

seq(from = 4, by = 0.5, length = 25)
```

```
##  [1]  4.0  4.5  5.0  5.5  6.0  6.5  7.0  7.5  8.0  8.5  9.0  9.5 10.0 10.5 11.0
## [16] 11.5 12.0 12.5 13.0 13.5 14.0 14.5 15.0 15.5 16.0
```

2.4.3 rep()

Funkcie *rep*() jednoducho zopakuje zadané číslo, alebo vektor, x-krát.

```
rep(x = 1, times = 10)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1
```

```
rep(x = c(1, 2, 3), times = 3)
```

```
## [1] 1 2 3 1 2 3 1 2 3
```

2.5 Indexovanie

Indexovanie je v R-ku veľmi užitočný spôsob selektovania dát. Ide jednoducho o výber súboru dát, zo súboru dát. Indexujeme za použitia hranatých zátvoriek, ktoré bez medzery nalepíme k objektu, z ktorého chceme dáta vybrať. Najjednoduchšie sa to vysvetľuje ukážkou. A nezabúdajte, že R-ko začína od jednotky, nie od nuly. Aj keď nám, ekonómom neprogramátorom to asi ani nepríde divné.

```
# Vytvorme si obyčajný číselný vektor.

obycajny_vektor <- c(1:10)

obycajny_vektor
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
# Za použitia indexovania môžeme vybrať akúkoľvek hodnotu. Chceme vybrať prvú.

obycajny_vektor[1]
```

```
## [1] 1
```

```
# Alebo súbor hodnôt. Vyberieme prvú a poslednú hodnotu.

obycajny_vektor[c(1, 10)]
```

```
## [1]  1 10
```

Pravdepodobne by väčšina z vás napísala $obycajny\_vektor[1, 10]$, to by vám však vyhodilo chybu. Prečo je tomu tak plne pochopíte, keď si ukážeme matice. Aj keď to nie je nič zložité. Vektor si predstavme ako šípku, ktorá určuje smer. Je to teda, v našom prípade, reťazec čísel, jedna dlhá šnúra, ktorá nemá žiadne riadky ani stĺpce. R-ko však to, čo napíšeme do hranatých zátvoriek vníma ako:

$$[riadok, stĺpec]$$

$$[row, column]$$

> *Zo začiatku sa mi zvyklo pliesť, čo ide prvé. Zapamätal som si to ako RC autíčko. Také tie malé na ovládanie.*

Čiže prvý údaj predstavuje riadok, a druhý údaj stĺpec. Ak by sme napísali $[1, 10]$, R-ko by hľadalo v prvom riadku desiatu hodnotu. Do hranatých zátvoriek píšeme vlastne **súradnice**. V reťazci hodnôt máme ale iba reťazec hodnôt. (:D) Preto je potrebné použiť funkciu $c()$, aby R-ko vedelo, že má vyberať z reťazca hodnôt. Indexovanie je veeeľmi užitočné, a dá sa využiť veľmi kreatívne. Nám však stačí vedieť, čo to plus-mínus robí. Aby ste vedeli, čo sa deje, keď uvidíte hranaté zátvorky. O indexovaní si ešte povieme pri maticiach.

```
# Indexovaním môžeme aj odobrať hodnotu. Napr., ak chceme všetky okrem poslednej:

obycajny_vektor[-10]
```

```
## [1] 1 2 3 4 5 6 7 8 9
```

2.6 Iné typy vektorov

Vektory nemusia obsahovať len čísla. Môžu obsahovať napríklad aj **textové** alebo **logické** premenné. Existuje ešte aj štvrtý typ, **faktorové** vektory, ktorým sa ale nebudeme zaoberať.

2.6.1 Vektor textových premenných

```
# Pre vytvorenie vektora obsahujúceho textové reťazce, musí byť obsah ohraničený
# úvodzovkami "".

vector_characters <- c("c", "musíme", "používať", "stále", "ak", "chceme",
                       "viac", "hodnôt")

vector_characters
```

```
## [1] "c"        "musíme"   "používať" "stále"    "ak"       "chceme"   "viac"
## [8] "hodnôt"
```

Vektor tvorený textovým reťazcom nájde svoje uplatnenie napríklad pri pomenovaní hodnôt.

```
nazvy <- c("prvy", "druhy", "treti")
cisla <- c(1, 2, 3)

# Použijeme na to funkciu names()

names(cisla) <- nazvy

# Ak sa teraz pozrieme na vektor "cisla", uvidíme, že sme číslam priradili názvy.
# Na vypísanie výsledku môžeme použiť aj funkciu print().

print(cisla)
```

```
##  prvy druhy treti
##     1     2     3
```

Vektor **“cisla”** sme vlozili do funkcie **names()**. Aplikovali sme funkciu na vektor, ktorému sme chceli priradiť názvy. **Priradiť**, teda symbol priradenia **<-**, potom už len vektor s názvami, ktoré chceme priradiť. Pre lepšiu ilustráciu si to napíšeme nanovo, bez zadefinovaného vektora “nazvy”.

```
names(cisla) <- c("adin", "dos", "tres")

print(cisla)
```

```
## adin  dos tres
##    1    2    3
```

2.6.2 Vektor logických premenných

```
# Logické operátory, inak známe ako Booleovské operátory, nám ako výsledok
# poskytnú výstup v podobe TRUE alebo FALSE.
# ! Pre overenie rovnosti použijeme "==".

vektor <-  5

vektor == 5
```

```
## [1] TRUE
```

```
vektor == 6
```

```
## [1] FALSE
```

Logický operátor	Popis
<	menšie než
<=	menšie alebo rovné
>	väčšie než
>=	väčšie alebo rovné
==	rovná sa
!=	nerovná sa
!x	nie je x
x	y
x & y	x a y
isTRUE(x)	test či je x pravdivé

Logické operátory sa zídu pri indexovaní, alebo pri zisťovaní počtu vyhovujúcich hodnôt.

```
# Vektor výšky ľudí, ktorý sme si skôr vytvorili.

vyska <- sample(x = 160:190, size = 10, replace = TRUE)

# Použitie logického operátora na zistenie, kto má viac ako 170cm.

viac_ako_170 <- vyska > 170

# Výsledky však nebudú číselnými hodnotami, ale hodnotami booleovského typu.

print(viac_ako_170)
```

```
##  [1] FALSE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE
```

```r
# To nám však nebráni zužiťkovať to pomocou funkcie "sum()" a zistiť počet
# vyhovujúcich hodnôt. Zráta to všetky TRUE hodnoty.

sum(viac_ako_170)
```

```
## [1] 5
```

```r
# Čo dokážeme pomocou indexovania pretvoriť na číselné hodnoty.

vyska_v_cm <- vyska[vyska > 170]

print(vyska_v_cm)
```

```
## [1] 186 173 189 178 173
```

2.7 Matice

#nainštalovat len jeden balik a potom ked prejdeme vektory tak c(viac balikov)

2.8 Import údajov

Table Header

Fukncia	Second Header
Content Cell	Content Cell
Content Cell	Content Cell