Week 6 - Codebook

November 12, 2024

1 Week 6: Codebooks & Cloud Computing

- Imagine this: you are traveling all you have is your data on Dropbox, Google Drive or your flashdrive. And you have the hotel's business center. What do you do? Hmmm...
- Sometimes, you need extra flexibility of running your code from any computer, even those that don't have all your software (R studio) on it.
- This codebook is a Jupyter Notebook and you are using Google's cloud services to run R code in it. Note: some advanced usage instances require Google's subscription services. You can similarly run Python code if you went into the 'Runtime' option above and select 'Change Runtime.'
- This also makes it easy to share code that is interactable and executable by anybody.

1.1 Clear the Workspace & Load Libraries

Note that not all packages are available on the cloud, so you might need to install some packages within an instance. Any data or package you call will only be available in that session. Once you close out the session you will lose these packages or data. So you will need to re-load them each time.

[1]:
```
suppressMessages(install.packages("emmeans"))
suppressMessages(install.packages("broom"))
```

[3]:
```
rm(list = ls())

ReqdLibs = c("ggplot2","dplyr","emmeans","broom")
invisible(suppressMessages(lapply(ReqdLibs, library, character.only = TRUE)))
```

1.2 Set theme for a figure

[8]:
```
thm = theme(
          plot.title = element_text(colour = "black",size = 35, face = "bold",␣
 ↪hjust = 0.5),
          legend.text = element_text(size = 18),
          legend.title = element_text(size = 20),
          axis.ticks.length = unit(0.3,"cm"),
          axis.line = element_line(colour = "black",linewidth = 1),
```

```
          axis.ticks = element_line(colour = "black",linewidth = 1),
          axis.text = element_text(colour = "black",size = 35),
          axis.text.x = element_text(lineheight = 1.1, margin = margin(t = 20)),
          axis.title.x = element_text(size=35, colour = "grey35", face =␣
 ↪"plain",
                                      lineheight = 1.1, margin = margin(r = 10)),
          axis.title.y = element_text(size=35, colour = "grey35", face =␣
 ↪"plain",
                                      lineheight = 1.1, margin = margin(r = 10)))
```

1.3 Let's load a built-in R dataset

[ ]:
```
head(diamonds)
```

A tibble: 6 × 10

carat <dbl>	cut <ord>	color <ord>	clarity <ord>	depth <dbl>	table <dbl>	price <int>	x <dbl>	y <dbl>	z <dbl>
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
0.29	Premium	I	VS2	62.4	58	334	4.20	4.23	2.63
0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
0.24	Very Good	J	VVS2	62.8	57	336	3.94	3.96	2.48

1.4 Use DPLYR functions to manipulate your data as needed

[ ]:
```
diamonds %>%
  filter(cut!="Fair") %>%
  select(clarity,cut, depth) %>%
  {.->>diamonds.ideal}

head(diamonds.ideal)
```

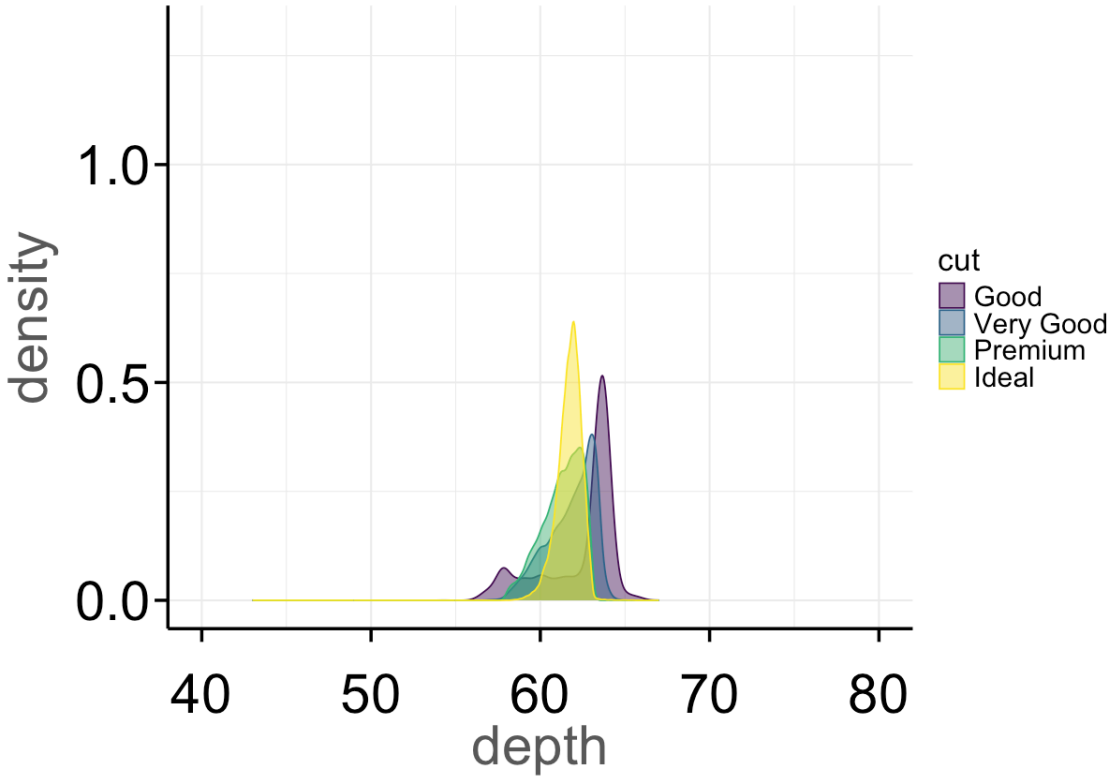
A tibble: 6 × 3

clarity <ord>	cut <ord>	depth <dbl>
SI2	Ideal	61.5
SI1	Premium	59.8
VS1	Good	56.9
VS2	Premium	62.4
SI2	Good	63.3
VVS2	Very Good	62.8

1.5 Plot the in-built data

[ ]:
```
options(repr.plot.width = 10, repr.plot.height = 7)

ggplot(diamonds.ideal, mapping = aes(x = depth, y = after_stat(density),
                                     group = cut, col = cut, fill = cut)) +
  # geom_histogram(bins = 50, fill = NA) +
```

```
  geom_density(alpha = 0.5) +
  # facet_wrap(~clarity) +
  coord_cartesian(xlim = c(40,80), ylim = c(0, 1.3)) +
    theme_minimal() + thm
```



1.6 How to load your own dataset?

Use the options bar on the left to upload data file in the temporary session.

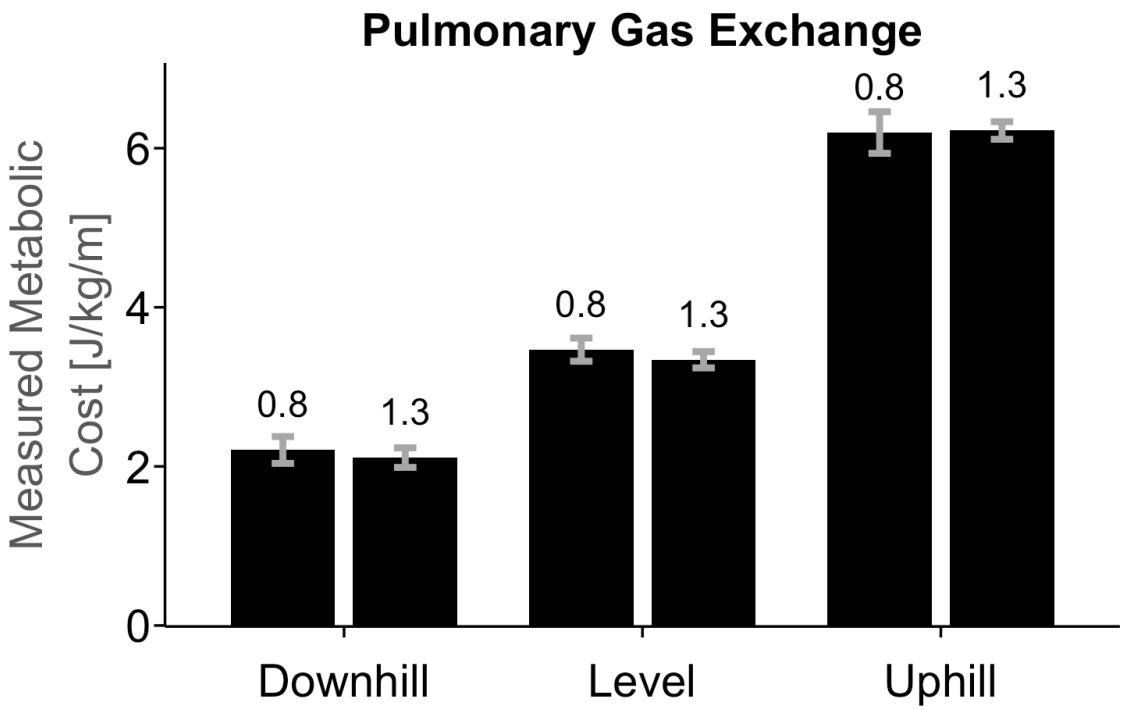
[6]:
```
dat.calc2 = read.csv("calcData.csv", header = TRUE)
head(dat.calc2)
```

A data.frame: 6 × 20

	Sub <chr>	cond <chr>	incline <chr>	speed <dbl>	R <dbl>	VT <dbl>	VE <dbl>	VO2 <dbl>	a
1	Sub1	walk	downhill	0.8	0.9019424	0.7859465	20.33910	728.5032	2
2	Sub1	walk	downhill	1.3	0.8661119	0.8409442	25.83893	1005.2468	2
3	Sub1	walk	level	1.3	0.8340541	1.0104406	32.61279	1368.0726	2
4	Sub1	walk	uphill	0.8	0.8661413	1.2505480	40.73454	1687.3811	2
5	Sub1	walk	uphill	1.3	0.8315770	1.4630620	58.84244	2540.5691	2
6	Sub1	walk	level	0.8	0.8727268	1.0036957	26.00091	992.9288	2

1.7 Plot your own data!

[9]:
```
options(repr.plot.width = 12, repr.plot.height = 8)

repro.fig =
# FUNCTION CALL
ggplot(dat.calc2, aes(x = incline,y = C_meas, group = speed, label = speed)) +

# LAYERS THAT SUMMARIZE WHILE PLOTTING! - this is one of the most powerful␣
 ↪features of ggplot
stat_summary(geom = "bar", fun = mean, col = NA, fill = "black", width = 0.7,␣
 ↪na.rm = TRUE,
             position=position_dodge(width = 0.82, preserve = 'single')) +
stat_summary(geom = "errorbar",fun.data = mean_se, width = 0.15, lwd=2.5,␣
 ↪col="darkgray", na.rm = TRUE,
            position=position_dodge(0.82, preserve = 'single')) +
stat_summary(geom = "text", fun = mean, position = position_dodge(0.82,␣
 ↪preserve = 'total'), na.rm = TRUE,
             vjust = -1.25, size = 10) +

# AXIS LIMITS
coord_cartesian(ylim = c(0.31,6.75)) +

# LABELS
scale_x_discrete(labels = c("Downhill", "Level", "Uphill")) +  # capitalize␣
 ↪label initials :/
labs(title = "Pulmonary Gas Exchange", x = "", y = "Measured Metabolic\nCost [J/
 ↪kg/m]") +
theme_classic() + thm

repro.fig
```



1.8 Run some statistics

In this case, we'll compare:

1. across the 3 inclines for each level of speed
2. between the 2 speeds for each level of incline

[23]:
```
mod.inter = lm(data = dat.calc2, W_adj ~ incline*speed)
# summary(mod.inter)

# below functions tidy and glance from the broom package allow you to view
# model outputs as tibbles, although they are not publication ready
tidy(mod.inter)
glance(mod.inter)
emm_inter = emmeans(mod.inter, ~incline|speed)
pairs(emm_inter, by = "incline") %>% tidy
```

A tibble: 6 × 5

term <chr>	estimate <dbl>	std.error <dbl>	statistic <dbl>	p.value <dbl>
(Intercept)	0.20006678	0.4675560	0.42789907	6.701390e-01
inclinelevel	0.06448867	0.6571845	0.09812873	9.221321e-01
inclineuphill	-0.25379433	0.6571845	-0.38618431	7.006214e-01
speed	1.95654666	0.4374894	4.47221515	3.179998e-05
inclinelevel:speed	1.18061479	0.6119414	1.92929375	5.806193e-02
inclineuphill:speed	4.30761814	0.6119414	7.03926535	1.478566e-09

A tibble: 1 × 12

r.squared <dbl>	adj.r.squared <dbl>	sigma <dbl>	statistic <dbl>	p.value <dbl>	df <dbl>	logLik <dbl>	AIC <dbl>
0.945088	0.940864	0.5240353	223.7425	1.486843e-39	5	-51.73032	117.4606

A tibble: 3 × 9

incline <chr>	term <chr>	contrast <chr>	null.value <dbl>	estimate <dbl>	std.error <dbl>	df <dbl>	statistic <dbl>
downhill	speed	speed0.8 - speed1.3	0	-0.9782733	0.2187447	65	-4.47221
level	speed	speed0.8 - speed1.3	0	-1.5685807	0.2139365	65	-7.33199
uphill	speed	speed0.8 - speed1.3	0	-3.1320824	0.2139365	65	-14.6402

That's all! Happy coding and sharing your code with the world!

1.9 # The End