

WEEK 3: CLEAN AND MERGE DATA

In Week 2, we combined individual raw data files into one comprehensive table. Now, we will merge this table with additional data, such as demographics, to prepare for analysis.

Currently, the table is in **long** format: time points are nested within each trial, and trials are nested within each subject (see figure for wide vs long formats).

This lesson will cover:

1. Basic data cleaning
2. Converting between long and wide data formats
3. Merging with a separate demographic table
4. Aggregating the data table
5. Using plots for data validation

Wide Format				Long Format		
Team	Points	Assists	Rebounds	Team	Variable	Value
A	88	12	22	A	Points	88
B	91	17	28	A	Assists	12
C	99	24	30	A	Rebounds	22
D	94	28	31	B	Points	91
				B	Assists	17
				B	Rebounds	28
				C	Points	99
				C	Assists	24
				C	Rebounds	30
				D	Points	94
				D	Assists	28
				D	Rebounds	31

Courtesy of [Statology](#).

CLEAN THE ENTIRE WORKSPACE

In []:

```
rm(list=ls())
```

LOAD REQUIRED LIBRARIES

In []:

```
ReqdLibs =
c("here", "ggplot2", "ggthemes", "dplyr", "tidyr", "corrplot", "readxl", "IRdisplay")
invisible(lapply(ReqdLibs, library, character.only = TRUE))
```

THEME DEFAULTS

In []:

```
thm = theme(
  strip.text.x=element_text(size=20, face="bold"),
  strip.text.y=element_text(size=20, face="bold"),
  legend.text=element_text(size=16, face="bold"),
  legend.position = "top",
  legend.title=element_text(size=16, face="bold"),
  title =element_text(size=14, face='bold'),
  text = element_text(colour = "black", size=18),
  plot.title = element_text(colour = "black", size = 22, face = "bold"),
  axis.ticks.length = unit(0.3, "cm"),
  axis.line = element_line(colour = "black", linewidth = 0.85),
  axis.ticks = element_line(colour = "black", linewidth = 0.85),
  axis.text = element_text(colour = "black", size=24),
  axis.title=element_text(size=25))
```

READ DATA

I previously saved the parent data set as a .csv I am now reading it in here.

In []:

```
data.all=read.csv("raw.data.all.csv")
head(data.all)
# data.all
```

A data.frame: 6 × 8

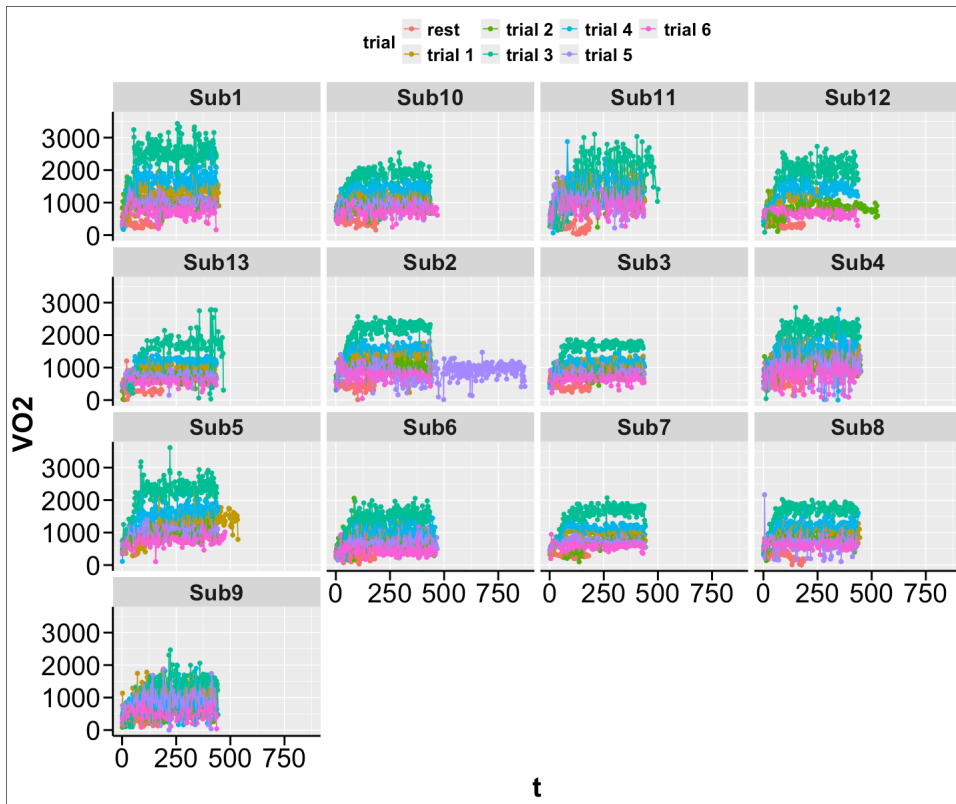
	X	t	Rf	VT	VE	VO2	Sub	trial
	<int>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>	<chr>
1	1	2	18.51852	0.7548531	13.97876	408.6987	Sub1	rest
2	2	5	20.97902	0.5538989	11.62026	304.7868	Sub1	rest
3	3	8	18.23708	0.7058896	12.87337	347.7043	Sub1	rest
4	4	10	25.53191	0.7823950	19.97604	614.8135	Sub1	rest
5	5	13	21.97802	0.4957548	10.89571	253.6431	Sub1	rest
6	6	16	18.92744	0.7874953	14.90527	437.0967	Sub1	rest

START BY VISUALIZING

Plots across the time series by subject

In []:

```
options(repr.plot.width = 12, repr.plot.height = 10)
ggplot(data.all, aes(x=t, y=VO2, color=trial)) +
  geom_point() +
  geom_line() +
  facet_wrap(~Sub) + thm
```



SOME INITIAL ISSUES

ISSUE 1: R NAMING CONVENTION

R still thinks my subjects are listed based on the value of the first integer. We need to change this first so it displays numerically.

In []:

```
display_markdown("**Note that at present the `Sub` variable is of character class.**  
So, when you look for the levels of this variable, you see the output as being NULL,  
as character class does not have levels.**")
class(data.all$Sub)
levels(data.all$Sub)
display_markdown("**But I can convert it to a factor.** This factor now has levels,  
and I can set the ordering for this variable as I want them to be.**")
data.all$Sub=factor(data.all$Sub, levels = c("Sub1", "Sub2", "Sub3", "Sub4",  
                                             "Sub5", "Sub6", "Sub7", "Sub8",  
                                             "Sub9", "Sub10", "Sub11", "Sub12",  
                                             "Sub13"))
#This is more desirable anyway because a factor class is treated as a categorical  
variable
# when you run regressions or other statistical analyses.
class(data.all$Sub)
levels(data.all$Sub)
```

Note that at present the `Sub` variable is of character class. So, when you look for the levels of this variable, you see the output as being NULL, as character class does not have levels.**

```
'character'
```

```
NULL
```

But I can convert it to a factor. This factor now has levels, and I can set the ordering for this variable as I want them to be.**

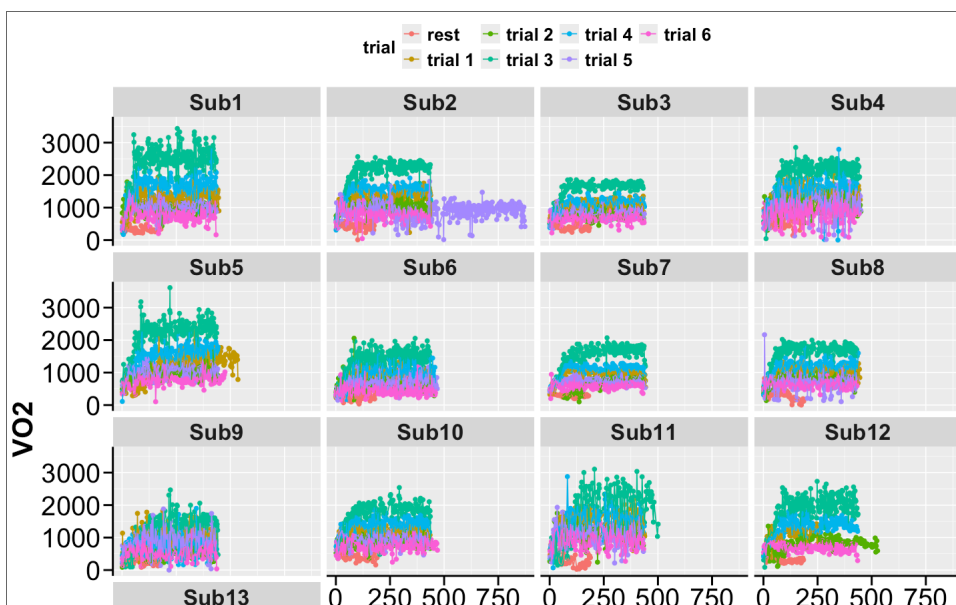
```
'factor'
```

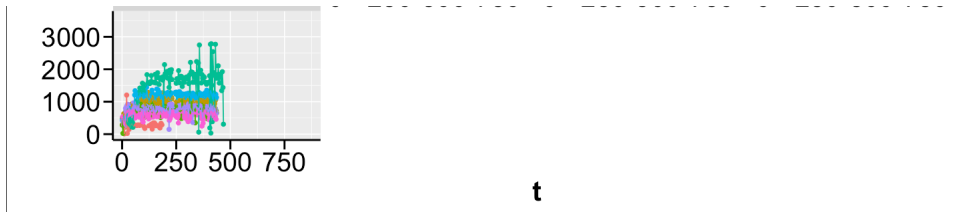
```
'Sub1' · 'Sub2' · 'Sub3' · 'Sub4' · 'Sub5' · 'Sub6' · 'Sub7' · 'Sub8' · 'Sub9' · 'Sub10' ·  
'Sub11' · 'Sub12' · 'Sub13'
```

VISUALIZE AGAIN AFTER FIXING THIS ISSUE

```
In [ ]:
```

```
#Fixed  
ggplot(data.all,aes(x=t,y=VO2,color=trial))+  
  geom_point()+  
  geom_line()+  
  facet_wrap(~Sub) + thm
```





ISSUE 2: INCONSISTENT NUMBER OF POINTS ACROSS TRIALS

Sub 2 took way more time than everyone else on trial 5. I am going to assume that doesn't warrant throwing that subjects data out, and instead I will just trim the excess away.

To do that, let's first see how much of the extra to remove. Let's look at how long all subject's trial 5 was. The function aggregate will provide some summary stat based on how I want the dataset aggregated. Here I just want the max time for trial 5 across all subjects. To reduce the output I just index the data.all to only include trial 5. Looks like Sub 2 did 873 seconds, and everyone else did ~440

In []:

```
aggregate(t~Sub,data.all[data.all$trial=="trial 5",],max) #>% mutate(a = median(.$t))
```

A data.frame: 12 × 2

Sub	t
<fct>	<int>
Sub1	436
Sub2	873
Sub3	437
Sub4	449
Sub5	443
Sub6	468
Sub7	441
Sub8	433
Sub9	433
Sub10	436
Sub11	433
Sub13	435

TRIM THE DATA

I am going to remove all t for Sub 2 greater than 440. I am going to use a method

called negation `!()`. Here I put in a bunch of boolean arguments into `()` and then preceded it with a `!`. This removes all the rows in my dataset that match these conditions.

Note that if I removed the `!()` it would only return the trial 5, for Sub2 with `t > 440`

In []:

```
display_markdown("**using base R**")
# using base R
data.all2 = data.all[!(data.all$trial=="trial 5" & data.all$Sub=="Sub2" &
data.all$t>440),]
head(data.all2)

display_markdown("**using dplyr**")
# using dplyr
data.all2 <-
data.all %>%
  filter(!(trial=="trial 5" & Sub=="Sub2" & t>440))
head(data.all2)
```

using base R

A data.frame: 6 × 8

	X	t	Rf	VT	VE	VO2	Sub	trial
	<int>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<fct>	<chr>
1	1	2	18.51852	0.7548531	13.97876	408.6987	Sub1	rest
2	2	5	20.97902	0.5538989	11.62026	304.7868	Sub1	rest
3	3	8	18.23708	0.7058896	12.87337	347.7043	Sub1	rest
4	4	10	25.53191	0.7823950	19.97604	614.8135	Sub1	rest
5	5	13	21.97802	0.4957548	10.89571	253.6431	Sub1	rest
6	6	16	18.92744	0.7874953	14.90527	437.0967	Sub1	rest

using dplyr

A data.frame: 6 × 8

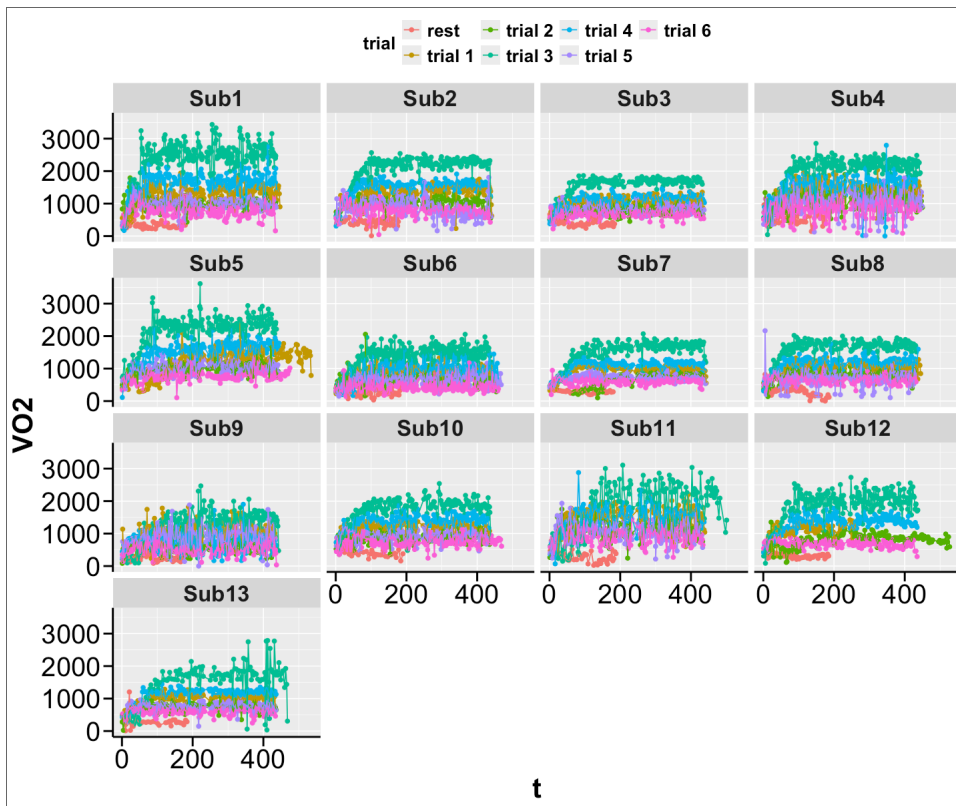
	X	t	Rf	VT	VE	VO2	Sub	trial
	<int>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<fct>	<chr>
1	1	2	18.51852	0.7548531	13.97876	408.6987	Sub1	rest
2	2	5	20.97902	0.5538989	11.62026	304.7868	Sub1	rest
3	3	8	18.23708	0.7058896	12.87337	347.7043	Sub1	rest
4	4	10	25.53191	0.7823950	19.97604	614.8135	Sub1	rest
5	5	13	21.97802	0.4957548	10.89571	253.6431	Sub1	rest
6	6	16	18.92744	0.7874953	14.90527	437.0967	Sub1	rest

VISUALIZE AGAIN AFTER FIXING THE ISSUE

In []:

```
#Fixed
ggplot(data.all2,aes(x=t,y=VO2,color=trial))+
  geom_point()+
  geom_line()+
  facet_wrap(~Sub) + thm
```

#I could repeat this process again but let's move on.



PIVOTING: CONVERTING DATA TABLES FROM LONG TO WIDE FORMAT AND VICE VERSA

Now I am going to convert the data from long to wide format using the `pivot` functions in the `tidyr` library. This can be useful if I want to look at correlations between my behavioral variables.

There are various situations where one format is preferable to others. **Note:** when in doubt, remember the tidy data format; each row is an independent observation and each column is a variable of interest. You should never stack numbers along rows of a column that do not share the same units.

FIRST, I AM GOING TO CREATE AN AGGREGATE OF THE DATA

This creates a summarized data set across our 4 outcome variables average across time for each subject for each trial

In []:

```
# using base R:
display_markdown("**using base R**")
data.agg = aggregate(cbind(V02,Rf,VE,VT) ~ trial + Sub, data.all2, mean)
head(data.agg)

# using dplyr:
display_markdown("**using dplyr**")
data.all2 %>%
  group_by(Sub, trial) %>%
  summarize(across(c(V02, Rf, VE, VT), mean), .groups = "drop") -> data.agg
head(data.agg)
dim(data.agg)
```

using base R

A data.frame: 6 x 6						
trial	Sub	VO2	Rf	VE	VT	
<chr>	<fct>	<dbl>	<dbl>	<dbl>	<dbl>	
1	rest	Sub1	340.4917	18.01353	12.45402	0.6969207
2	trial 1	Sub1	1303.5808	30.71848	30.52191	0.9942740
3	trial 2	Sub1	995.2381	26.19241	26.57332	1.0231692
4	trial 3	Sub1	2406.8231	39.76738	54.23449	1.3645677
5	trial 4	Sub1	1630.8077	32.55784	39.19033	1.2078367
6	trial 5	Sub1	996.5392	31.26921	25.57869	0.8297402

using dplyr

A tibble: 6 × 6					
Sub	trial	VO2	Rf	VE	VT
<fct>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>
Sub1	rest	340.4917	18.01353	12.45402	0.6969207
Sub1	trial 1	1303.5808	30.71848	30.52191	0.9942740
Sub1	trial 2	995.2381	26.19241	26.57332	1.0231692
Sub1	trial 3	2406.8231	39.76738	54.23449	1.3645677
Sub1	trial 4	1630.8077	32.55784	39.19033	1.2078367
Sub1	trial 5	996.5392	31.26921	25.57869	0.8297402

89 · 6

NEXT, LET'S "PIVOT" THE DATA FRAME TO A WIDE FORMAT

along the trial variable

```
In [ ]:
```

[illegible]

```
head(data.agg.wide)
```

Sub	VO2_rest	VO2_trial 1	VO2_trial 2	VO2_trial 3	VO2_trial 4	VO2_trial 5	VO
<fct>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	
Sub1	340.4917	1303.5808	995.2381	2406.823	1630.8077	996.5392	756
Sub2	425.3084	1323.6394	1020.4479	2078.080	1496.5663	871.1348	784
Sub3	420.5079	1051.5241	784.2673	1573.928	1092.0823	826.5741	681
Sub4	551.1862	1386.2685	1138.2881	2072.835	1515.1282	1045.7354	821
Sub5	NA	1240.8729	1010.5470	2211.938	1523.3187	1021.5176	760
Sub6	260.5606	859.7148	672.5981	1380.404	905.8012	668.0126	443

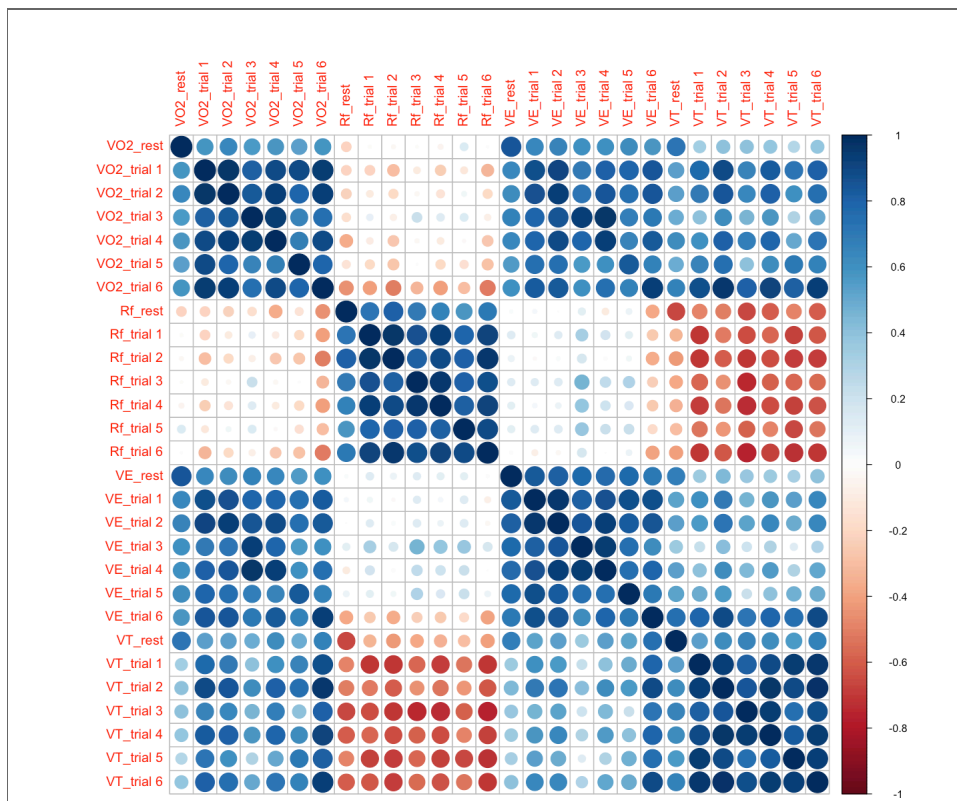
VISUALIZING CORRELATIONS ACROSS TRIALS USING THE WIDER FORMAT DATA

In []:

```
#create a correlation matrix with only complete observations,
#Can only include numeric variables.
cmat=cor(data.agg.wide[, -1], use="complete.obs")

#Let's visualize the correlations.
corrplot(cmat) + theme_minimal() + thm
```

NULL



NOW, LET'S PIVOT BACK TO THE LONG FORMAT

In []:

```
# Now let's convert our wide dataset back to long, just for fun.
data.agg.long <- pivot_longer(data.agg.wide,
                              cols = V02_rest:`VT_trial` 6,
                              names_to = c(".value", "trial"),
                              names_sep = "_")
head(data.agg.long)
```

A tibble: 6 × 6

Sub	trial	VO2	Rf	VE	VT
<fct>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>
Sub1	rest	340.4917	18.01353	12.45402	0.6969207
Sub1	trial 1	1303.5808	30.71848	30.52191	0.9942740
Sub1	trial 2	995.2381	26.19241	26.57332	1.0231692
Sub1	trial 3	2406.8231	39.76738	54.23449	1.3645677
Sub1	trial 4	1630.8077	32.55784	39.19033	1.2078367
Sub1	trial 5	996.5392	31.26921	25.57869	0.8297402

MERGE WITH DEMOGRAPHICS

Now that we have our data in a variety of formats we can now merge it with the demographic data.

In []:

```
demo = read_excel("SubjectInfo.xlsx")
head(demo)
```

A tibble: 6 × 8

Subject No	Age	Reported Weight (kg)	Reported Length (cm)	Gender	Level Slow	Level Walk	Weight from force plates(kg)
<chr>	<dbl>	<dbl>	<dbl>	<chr>	<dbl>	<dbl>	<dbl>
Sub1	26	86	185	M	885.5529	891.5307	90.57511
Sub2	28	77	178	F	767.7686	760.2377	77.88004
Sub3	21	52	170	M	530.6408	558.2018	55.49656
Sub4	25	73	168	M	NA	NA	NA
Sub5	34	86	173	M	878.6303	898.5188	90.57845
Sub6	19	54	160	F	553.4936	558.2845	56.66555

MERGING THE DEMO DATA TO THE AGGREGATE DATA IN THE LONG FORMAT

Note that if the names of the reference variable (in this case subject ID) by which the two data frames are being merged are different then we have to call that explicitly.

In []:

```
demo.data = merge(data.agg,demo,by.x = "Sub",by.y = "Subject No")
head(demo.data)
```

A data.frame: 6 × 13

	Sub	trial	VO2	Rf	VE	VT	Age	Reported Weight (kg)	Reported Length (cm)
	<fct>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	Sub1	rest	340.4917	18.01353	12.45402	0.6969207	26	86	181
2	Sub1	trial_1	1303.5808	30.71848	30.52191	0.9942740	26	86	181
3	Sub1	trial_2	995.2381	26.19241	26.57332	1.0231692	26	86	181
4	Sub1	trial_3	2406.8231	39.76738	54.23449	1.3645677	26	86	181
5	Sub1	trial_4	1630.8077	32.55784	39.19033	1.2078367	26	86	181
6	Sub1	trial_5	996.5392	31.26921	25.57869	0.8297402	26	86	181

MERGING THE DEMO DATA TO THE AGGREGATE DATA IN THE WIDE FORMAT

See how the demographic variables in the table above repeat themselves over the different trials for each subject. This is because the `merge()` function identifies that each subject has only 1 number for weight or height but it has 7 rows of trial-wise data. So it repeats them over these 7 rows. What type of table would be ideal to merge with the demographic table?...where all of the trials for one subject are in a single row? The wide format!

So, let's combine the demographic data with our previously created `data.agg.wide` dataframe.

In []:

```
demo.data_wide = merge(data.agg.wide,demo,by.x = "Sub",by.y = "Subject No")
head(demo.data_wide)
```

	Sub	VO2_rest	VO2_trial 1	VO2_trial 2	VO2_trial 3	VO2_trial 4	VO2_trial 5	
	<fct>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	
1	Sub1	340.4917	1303.5808	995.2381	2406.823	1630.808	996.5392	7
2	Sub10	430.0402	1181.1066	879.9850	1776.831	1350.246	929.5444	7
3	Sub11	299.8132	1441.6617	1114.7714	1836.072	1502.942	1039.9077	9
4	Sub12	318.8275	1100.9149	795.6721	1901.475	1332.632	NA	6
5	Sub13	289.7562	986.0555	732.9015	1458.078	1126.762	733.7484	5
6	Sub2	425.3084	1323.6394	1020.4479	2078.080	1496.566	871.1348	7

NOW LET'S PLOT ONE OF THE DEMOGRAPHIC VARIABLES AGAINST ONE OF THE METABOLIC VARIABLES

We're plotting weight against VO2 from trial 3. You see how a wide data format can be useful for correlations and scatterplots of this type where 1 row has data unique to 1 subject.

In []:

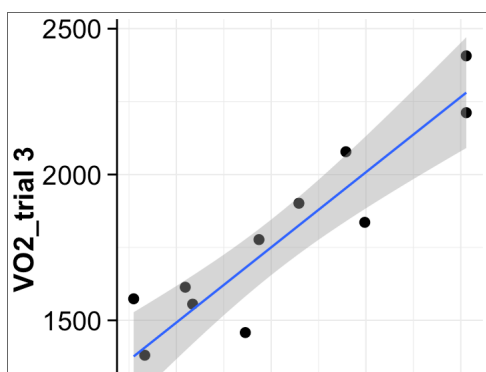
```
options(repr.plot.width = 6, repr.plot.height = 6)
ggplot(data = demo.data_wide, aes(x = `Weight from force plates(kg)`, y = `VO2_trial
3`)) +
geom_point(size = 4) + geom_smooth(method = "lm", formula = y~x) + theme_minimal() +
thm
```

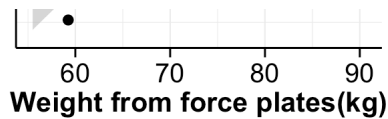
Warning message:

"Removed 1 row containing non-finite outside the scale range (`stat_smooth`
h())."

Warning message:

"Removed 1 row containing missing values or values outside the scale range
(`geom_point()`)."





LET'S PIVOT THE AGGREGATE DATA AGAIN TO LOOK AT ALL VARIABLES IN ONE COLUMN

In []:

```
demo.data.var.long = pivot_longer(demo.data,
                                   cols = V02:VT,
                                   names_to = "Variable",
                                   values_to = "Value")

head(demo.data.var.long)
dim(demo.data.var.long)
```

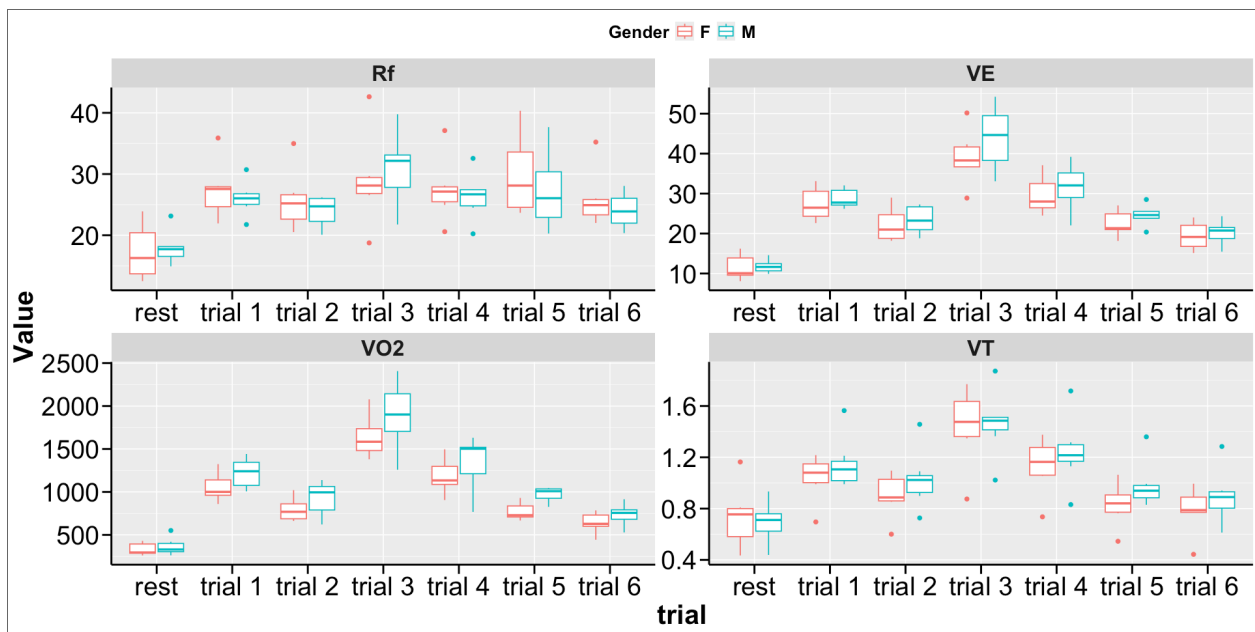
A tibble: 6 × 11

Sub	trial	Age	Reported Weight (kg)	Reported Length (cm)	Gender	Level Slow	Level Walk	Weight from force plates(kg)	Variable
<fct>	<chr>	<dbl>	<dbl>	<dbl>	<chr>	<dbl>	<dbl>	<dbl>	<chr>
Sub1	rest	26	86	185	M	885.5529	891.5307	90.57511	VO2
Sub1	rest	26	86	185	M	885.5529	891.5307	90.57511	Rf
Sub1	rest	26	86	185	M	885.5529	891.5307	90.57511	VE
Sub1	rest	26	86	185	M	885.5529	891.5307	90.57511	VT
Sub1	trial 1	26	86	185	M	885.5529	891.5307	90.57511	VO2
Sub1	trial 1	26	86	185	M	885.5529	891.5307	90.57511	Rf

356 · 11

In []:

```
options(repr.plot.width = 16, repr.plot.height = 8)
ggplot(demo.data.var.long, aes(x=trial, Value, color=Gender))+
  geom_boxplot()+
  facet_wrap(~Variable, scales = "free") + thm
```



THE END