Python ↔ R Data Wrangling Cheat Sheet (pandas ↔ dplyr/tidyr)

Assume: R uses dplyr/tidyr; Python uses pandas. Snippets are minimal + copy-paste friendly.

Setup (tiny example data)

R

```r
library(dplyr)
library(tidyr)

df <- tibble(
  name = c("A","B","C","C"),
  age  = c(25, 30, 22, 22),
  score= c(88, 92, 95, 95),
  grp  = c("g1","g1","g2","g2")
)
```

Python

```python
import pandas as pd

df = pd.DataFrame({
  "name": ["A","B","C","C"],
  "age":  [25, 30, 22, 22],
  "score":[88, 92, 95, 95],
  "grp":  ["g1","g1","g2","g2"]
})
```

Select columns

R	Python
`df %>% select(name, score)`	`df[["name", "score"]]`
`df %>% select(starts_with("s"))`	`df.loc[:, df.columns.str.startswith("s")]`

Filter rows

R	Python
`df %>% filter(age > 24, grp == "g1")`	`df[(df["age"] > 24) & (df["grp"] == "g1")]`

Create/transform columns (mutate)

R

```r
df %>%
  mutate(
    passed = score > 90,
    score_z = (score - mean(score)) / sd(score))
  )
```

Python

```python
df.assign(
  passed = df["score"] > 90,
  score_z = (df["score"] - df["score"].mean()) / df["
```

Group + summarize (aggregation)

R

```r
df %>%
  group_by(grp) %>%
  summarise(
    n = n(),
    mean_score = mean(score),
    max_age = max(age),
    .groups = "drop"
  )
```

Python

```python
df.groupby("grp").agg(
  n=("grp", "size"),
  mean_score=("score", "mean"),
  max_age=("age", "max")
).reset_index()
```

Sort (arrange)

R	Python
`df %>% arrange(desc(score), age)`	`df.sort_values(["score", "age"], ascending=[False, Tr`

Rename columns

R	Python
`df %>% rename(points = score)`	`df.rename(columns={"score": "points"})`

Distinct / drop duplicates

R	Python

`df %>% distinct()`
`df %>% distinct(name, .keep_all = TRUE)`
	`df.drop_duplicates()`
	`df.drop_duplicates(subset=["name"], keep="first")`

Join / merge

R	Python

`lookup <- tibble(grp=c("g1","g2"), label=c("Group 1","Group 2"))`
 	
`df %>% left_join(lookup, by = "grp")`
	`lookup = pd.DataFrame({"grp":["g1","g2"], "label":["G`
	
	`df.merge(lookup, on="grp", how="left")`

Reshape: pivot wider

R	Python

`# example: mean score by grp & name, wide by grp`
`df %>%`
`  group_by(name, grp) %>% summarise(mean_score=mean(score), .groups="drop") %>%`
`  pivot_wider(names_from = grp, values_from = mean_score)`
	`tmp = (df.groupby(["name","grp"], as_index=False)`
	`         .agg(mean_score=("score","mean")))`
	`tmp.pivot(index="name", columns="grp", values="mean_s`

Reshape: pivot longer (wide → long)

R	Python

`wide <- tibble(name=c("A","B"), g1=c(1,2), g2=c(3,4))`
 	
`wide %>%`
`  pivot_longer(cols = c(g1, g2), names_to="grp", values_to="value")`
	`wide = pd.DataFrame({"name":["A","B"], "g1":[1,2], "g`
	
	`wide.melt(id_vars=["name"], value_vars=["g1","g2"],`
	`          var_name="grp", value_name="value")`

Handle missing values

R	Python

`df2 <- tibble(x=c(1, NA, 3), y=c("a", NA, "c"))`
 	
`df2 %>% drop_na()`
`df2 %>% replace_na(list(x = 0, y = "missing"))`
	`import numpy as np`
	`df2 = pd.DataFrame({"x":[1, np.nan, 3], "y":["a", np.`
	
	`df2.dropna()`
	`df2.fillna({"x": 0, "y": "missing"})`

Split-apply-combine (within groups)

R	Python

`df %>%`
`  group_by(grp) %>%`
`  mutate(rank_in_grp = dense_rank(desc(score))) %>%`
`  ungroup()`
	`df.assign(`
	`  rank_in_grp = df.groupby("grp")["score"].rank(metho`
	`)`